Univerzita Karlova
Pedagogická fakulta
Katedra informačních technologií a technické výchovy

BAKALÁŘSKÁ PRÁCE

Programovací nástroje pro algoritmizaci a programováni na ZŠ

Programming tools for development of algorithms and programs at elementary school

Jan Dümont

Vedoucí práce:	Ing. Jaroslav Novák, Ph.D.
Studijní program:	Specializace v pedagogice
Studijní obor:	Informační technologie se zaměřením na vzdělávání

2018

Prohlašuji, že jsem bakalářskou práci na téma „Programovací nástroje pro algoritmizaci a programováni na ZŠ“ vypracoval pod vedením vedoucího bakalářské práce samostatně za použití v práci uvedených pramenů a literatury. Dále prohlašuji, že tato bakalářská práce nebyla využita k získání jiného nebo stejného titulu.

V Litoměřicích dne 5.12. 2018

…… podpis

Rád bych na tomto místě poděkoval vedoucímu práce, Ing. Jaroslavu Novákovi, Ph.D., za jeho cenné rady, odbornou pomoc a velkou trpělivost při vedení bakalářské práce. Dále bych chtěl poděkovat své přítelkyni Lence za nekonečnou trpělivost během psaní této práce.

ANOTACE

Tato bakalářská práce analyzuje minulý a současný stav vzdělávací oblasti algoritmizace a programování na základní škole na základě výzkumů prováděných na Katedře informačních technologií a technické výchovy Pedagogické fakulty UK v Praze. Přináší přehled různých výukových metod a programů, které napomáhají k rozvíjení dovedností žákům vzhledem k jejich věku a dosaženým dovednostem v této oblasti. V teoretické části práce jsou také vymezeny základní pojmy tématu algoritmizace a programování. Na ni navazuje praktická část využívající teoretických poznatků ve výuce informatiky i v jiných předmětech.

KLÍČOVÁ SLOVA

Algoritmus, informatické myšlení, propedeutika, programovací jazyk, robot

ABSTRACT

This thesis monitors the past and current status of the educational area of algorithms and computer programming in elementary school, on the basis of research carried out at the Department of Information Technology and Technical Education of the Pedagogical Faculty of Charles University in Prague. It provides an overview of the different teaching methods and programs that contribute to the development of the skills of pupils, due to their age and skills achieved in this area. In the theoretical part of the thesis are explained the basic concepts of the topic of algorithms and programming, which follows the practical part with the use of these procedures in the ICT lessons and in other school subjects.

KEYWORD

Algorithm, Computational Thinking, Propaedeutics, Programming Language, Robot

Obsah

1. Úvod

Dle závěrů z několika výzkumů, o kterých se budu v této práci ještě blíže zmiňovat, je *programování a algoritmické myšlení* považováno za okrajovou až zanedbatelnou informačně technologickou kompetenci. Na těchto závěrech se shodují dotazovaní učitelé informačních technologií na základních školách stejně jako dotazovaní žáci.

S blížícím se termínem završení nového konceptu vzdělávání informaticky zaměřených předmětů, který je podrobně rozpracován ve vládní studii nazvané *Strategie digitálního vzdělávání do roku 2020*, bude nezbytné rozšířit získávané informace také o základní poznatky z oblasti algoritmizace a programování, na které by posléze mělo navázat učivo střední školy, ve kterém budou takto získané poznatky dále rozšiřovány.

Pro získání prvních poznatků v této oblasti je velmi vhodné využít některých z mnoha dostupných propedeutických programovacích jazyků, popřípadě využít různých her nebo robotické techniky, kde lze kompetence algoritmického myšlení nacvičovat.

V této práci se zaměřím na představení a porovnání jednotlivých propedeutických jazyků, které jsou na našem trhu k dispozici, jak ve formě placeného softwaru, tak ve formě volně dostupného desktopového softwaru nebo online prostředí.

Jedním z výstupů práce budou pracovní listy a metodika pro pedagogy informaticky zaměřených předmětů, kteří je budou moci využít při rozvíjení kompetencí žáků v oblasti algoritmického myšlení a programování.

2. Cíle práce

Práce se zabývá vybranými metodami výuky algoritmizace a programování na základní škole. Cílem bakalářské práce je analýza možných využití alternativních přístupů podporujících výuku algoritmizace a programování vedoucí k rozvoji algoritmického myšlení žáků základní školy v souladu se Strategií digitálního vzdělání do roku 2020. Práce se opírá o informace z výzkumů o stavu výuky algoritmizace a programování na základních školách prováděných na katedře informatiky Pedagogické fakulty Univerzity Karlovy v Praze, a o výsledky vlastního průzkumu.

Práce reflektuje závěry a potřeby vládní směrnice Strategie digitálního vzdělávání do roku 2020. Tato směrnice zařazuje výuku programování a rozvíjení algoritmického myšlení nejen do čistě informatických předmětů, ale také do předmětů neinformatického charakteru.

Uvedené alternativní přístupy jsou porovnávány v rámci několika hledisek, kterými jsou převážně rozdělení dle věkové kategorie žáka a vhodnost dané metody k rozvíjení konkrétních kompetencí zaměřených na algoritmické myšlení. V rámci jedné alternativy je vždy zpracován jeden zástupce programovací pomůcky, programovacího prostředí nebo metody vedoucí k rozvoji informatického myšlení.

Studie obsahuje též empirickou část zaměřenou na ověření vhodnosti užití vybraného prostředí ve zvoleném typu edukace.

3. Vymezení hlavních pojmů

3.1 Algoritmus a algoritmizace

V dnešní přetechnizované době jsou počítače a výpočetní technika implementovány téměř do všech oblastí lidských činností, které jsou využívány k řešení rozmanitého spektra specifických úkolů. V převážné většině případů se k řešení těchto úkolů využívají předem připravené postupy, popisující jednotlivé kroky v jistém chronologickém pořadí, které nazýváme algoritmy.

Algoritmus je schematický postup pro řešení určitého druhu problémů, který je prováděn pomocí konečného množství přesně definovaných kroků. (Kvasil 1984)

Algoritmus je přesný postup, který popisuje řešení daného problému v konečném počtu kroků. (Pšeničková 2007)

Vzhledem k obecnému pojetí definice algoritmu, která je však dostačující pro účely prvotní představy o algoritmu, si pro úplnost uvedeme ještě klasickou programátorskou definici.

„Algoritmus je přesně určená konečná posloupnost instrukcí, jejichž realizace nám umožní pro přípustné hodnoty vstupních dat nalézt po konečném počtu kroků správná výstupní data.“ (Wirth 1989, cit v Havelková 2017, s. 5)

Jelikož se pojem algoritmus převážně spojuje s užitím výpočetní techniky, většině lidí přijde jejich tvorba velmi složitá a mají ji spojenou s odborným informačně technologickým vzděláním. Mnozí lidé, kteří zastávají tento názor, si však neuvědomují, že algoritmy bez využití výpočetní techniky používají ve svém každodenním životě. Většina v pasivní formě, kdy za předpokladu dodržení předepsaných kroků dosahují kýžených výsledků, ke kterým byl algoritmus vytvořen. Typickým příkladem algoritmu, ke kterému není třeba využití výpočetní techniky, je kuchařský recept, příručka typu „Udělej si sám“ nebo servisní manuál.

Použití takovýchto známých algoritmů předpokládá využití člověkem – tedy bytostí samostatně uvažující. Z tohoto důvodu mohou být součástí takového algoritmu i příkazy, které nejsou jednoznačné. Autor algoritmu, s ohledem na inteligenci člověka,

předpokládá jisté elementární všeobecné znalosti, a proto není nutné dílčí úkoly algoritmu blíže specifikovat.

Pokud je v návodu přikázáno *zašroubujte vrut A do otvoru B*, není třeba fázi šroubování blíže rozepisovat, tedy psát *uchopte šroubovák do pravé ruky (případně do levé, dle pravolevé orientace uživatele), nasaďte hrot na drážku, kterou naleznete na hlavici vrutu.* Tento přístup by velmi prodlužoval jednotlivé postupy a do jisté míry by zbytečně tříštil podstatné části daného algoritmu.

Jiným příkladem může být kuchařský recept na pečení sušenek, kdy je u přípravy těsta napsáno: *Do těsta přidejte cukr či sůl a důkladně prohněťte.* Volba použití příslušného dochucovadla je na uvážení uživatele a záleží na druhu využití. Není třeba v receptu dodávat, že pokud chceme podávat sušenky ke kávě, měli bychom těsto osladit a sůl zvolit v případě pečení sušenek podávaných např. k vínu.

Výše zmiňované pravidlo nelze použít při tvorbě algoritmu zpracovávaného strojem, neboť v tomto případě se nepředpokládá samovolné ovlivňování postupu inicializovaného strojem. Automat na pečení sušenek bude mít speciální algoritmus pro výrobu slaného pečiva, který bude odlišný od algoritmu na výrobu sladkých sušenek. Na rozdíl od člověka musí mít automat striktně zadány všechny ingredience, které do těsta přidá.

Dle Pšeničkové (2007) je třeba zajistit dodržení několika základních pravidel, která musí počítačový algoritmus splňovat:

- ***Determinovanost***: v každém kroku algoritmu musí být zcela jednoznačně určen krok následující.
- ***Obecnost***: abychom mohli daný postup označit jako algoritmus, nesmí se omezovat pouze na řešení jednoho specifického problému, ale jeho aplikace musí být použitelná pro obdobné problémy.
- ***Resultativnost***: každý algoritmus by po dosažení určitého počtu kroků měl vést k výsledku.
- ***Konečnost***: algoritmus řešící určitý problém má konečný počet kroků, který se odvíjí od složitosti struktury daného algoritmu a počtu vstupních dat. Existují algoritmy, které najdou řešení problému po několika málo krocích stejně tak jako

algoritmy, kde řešení problému může zabrat větší časový úsek. Tento algoritmus však musí po určitém počtu kroků skončit.

Tato pravidla neplatí v případě objektivně orientovaných programů, ve kterých není další krok jednoznačně dán, protože tento krok stanovuje uživatel ovládáním programu.

Algoritmizací v užším slova smyslu rozumíme cílený postup tvorby algoritmů, jež jsou následně zpracovány výpočetní technikou. (Bromová 2012, s. 5)

Celý proces můžeme rozdělit do pěti fází:

- ***Formulace problému*** **–** je první fází, ve které si stanovíme požadavky k řešení problému na základě vložených vstupních a výstupních hodnot požadovaných přesností výsledku.
- ***Analýza problému*** **–** je druhou fází algoritmizace, ve které na základě množství vstupních hodnot musíme rozhodnout, zda je problém řešitelný. V případě, že je problém řešitelný, snažíme se najít nejjednodušší řešení.
- ***Sestavení algoritmu pomocí formálního zápisu*** – je třetí fází, ve které naše řešení problému z druhé fáze přepíšeme do symbolické formy pomocí přesného pořadí dílčích operací, které vedou k řešení daného problému. Tento krok však zatím neřeší problém samotný, pouze naznačuje možný postup.
- ***Vytvoření programu pomocí programovacího jazyku*** **–** je předposlední fáze procesu algoritmizace, která převádí daný postup do konkrétního programovacího jazyka.
- ***Odladění programu*** **–** je poslední fází procesu, ve kterém dochází k následným odstraňování programových chyb.

Tyto chyby dělíme podle Bromové (2012, s. 5) do dvou základních skupin.

- ***Syntaktické chyby*** jsou méně závažnou formou chyb, ve kterých dochází k provinění proti řádnému zápisu určitého příkazu či procedury, způsobených překlepem nebo chybějící částí dílčí části zápisu. Syntaktické chyby jsou snadno odhalitelné díky překladači, který na podobné prohřešky upozorní, a zároveň označí jejich umístění v programovém kódu.

- ***Funkční chyby*** patří mezi kritické chyby, ke kterým dochází z důvodu špatné nebo nedostatečné analýzy problému a následnému sestavení vhodného algoritmu. Tento druh chyb se velmi těžko odhaluje, jeho výskyt se projevuje neočekávanými výstupními hodnotami, které nebyly v původním algoritmu stanoveny.

Hlavním cílem výuky algoritmizace a programování přitom není naučit děti programovat v konkrétním programovacím jazyku, ale naučit děti algoritmicky přemýšlet a daný problém analyzovat, tedy umět rozdělit danou úlohu na menší podúlohy, které žák dokáže vyřešit. (Pitner 2017)

3.2 Informatické myšlení

V zahraničí je pojem „Computational Thinking“ (dále už jen CT) úzce spojován se samostatným vyučujícím předmětem „Computer Science“, jež představuje zcela nový přístup k výuce informatiky. Doslovný český překlad tohoto termínu, který by zněl „výpočetní myšlení“, je srozumitelněji překládán jako „informatické myšlení“. V této kapitole bude nový termín podrobněji vysvětlen.

Hned na začátek je třeba uvést častou představu, že informatické myšlení je pouze doménou profesionálních programátorů a tudíž má smysl jen u žáků, kteří se takovému povolání chtějí v budoucnu věnovat. Tato představa plyne z neznalosti nebo nepochopení uvedeného termínu.

Informatické myšlení, stejně tak jako samotná algoritmizace, nemá za cíl vychovávat budoucí programátory, ale naopak má rozvíjet myšlenkové procesy při řešení problémů. Hlavní náplní tohoto procesu je naučit děti přemýšlet systémově. Tento přístup však není ničím převratným, neboť hodně lidí ho využívá nevědomě při každodenní aktivitě. Příkladem může být maminka, která ukládá zakoupené jogurty do lednice do prioritní fronty dle data spotřeby (Lessner 2018, s. 1).

Žák při balení školní brašny také projevuje informatické myšlení. Dalším příkladem je výběr nejrychlejší fronty v obchodě využívající modelování výkonnosti (Lessner 2018, s. 1).

Díky tomuto trendu by se měla změnit koncepce výuky informatiky od pouhého zvládání užití ICT ke kultivaci myšlení. K tomuto procesu by však nemělo docházet pouze při výuce informatiky, ale také v jiných předmětech, např. v matematice.

Poprvé použil pojem „informatické myšlení“ Saymour Papert, autor původního programovacího jazyku LOGO, když přiblížil možnost využití ICT ve výuce matematiky.

Diskuzi o informatickém myšlení rozšířila Jeanett Wingová, když ho přirovnala k základním schopnostem, jako jsou čtení, psaní a počítání a zdůraznila tak jeho potřebu ve výuce.

CT jsou myšlenkové postupy zapojené při takovém formulování problémů a jejich řešení, které umožňují tato řešení efektivně provést agentem zpracovávajícím informace. (Wing 2010)

Tímto agentem Janett Wingová rozumí stroj i člověka. Vymezení J. Wingové je příliš obecné, proto není vhodné pro aplikaci a školní prostředí.

Mnohem přesnější a konkrétnější definici přináší International Society for Technology in Education (ISTE) a Computer Science Teachers Association (CSTA). (Stephenson 2011)

CT dle Lessnera (2018, s. 3), je postup řešení problému, který zahrnuje mimo jiné následující charakteristiky.

- *Formulovat problémy způsobem, který umožňuje jejich strojové řešení.*
- *Logicky uspořádat a zkoumat data.*
- *Reprezentovat data prostřednictvím abstrakcí, jako jsou modely a simulace.*
- *Automatizovat myšlení pomocí algoritmického myšlení (jako posloupnost kroků).*
- *Odhalit, prozkoumat a provést možná řešení s cílem odhalit nejúčinnější kombinaci činností a zdrojů.*
- *Zobecňovat a přenášet tento postup řešení problémů do nejrůznějších dalších oblastí.*

Nedílnou součástí, bez kterých by se CT nemohlo rozvíjet, jsou samozřejmě také jisté předpoklady a postoje.(Lessner 2018, s. 3)

- *Sebejistota tváří v tvář složitosti.*
- *Vytrvalost při řešení obtížného problému.*

- *Snášení nejednoznačnosti.*
- *Schopnost vypořádat se s otevřenými problémy.*
- *Schopnost dorozumět se a spolupracovat s ostatními při dosahování společného cíle.*

Naprosto jednoduchou definici, dalo by se říci nevědeckou, přináší společnost Google, která v této oblasti vytváří vlastní web. (Lessner 2018, s. 3)

„CT zahrnuje sadu technik a dovedností k řešení problémů, které při psaní běžně používaných aplikací (vyhledávání, email, mapy), používají softwaroví inženýři. CT je nicméně využitelné téměř v jakémkoliv předmětu.

Součástí CT jsou zejména

- *rozklad problému.*
- *rozpoznávání vzorů (např. v grafech na burze, ale i v procesech).*
- *zobecňování vzorů (tedy vytváření abstraktních modelů).*
- *navrhování algoritmů.“*

3.3 Tvořivé myšlení

Tvořivost je zvláštní soubor schopností, které umožňují uměleckou, vědeckou nebo jinou tvůrčí činnost. Ta se projevuje jako vynalézavost, jako vznik nového, originálního díla nebo myšlenky, popř. tvůrčím řešením problémů. (Tvořivost 2018)

Tento tvůrčí proces lze podle Žáka (Žák 2004), který čerpal z primární teorie formulované sociálním psychologem a ekonomem Grahamem Wallasem, rozdělit do čtyř fází:

- *Příprava* – v této fázi dochází k pojmenovávání, analyzování a zpracování problémové situace.
- *Inkubace* – v této fázi získává řešitel od problému odstup. Oproti předchozí fázi, kde hrála roli spíše logika (analýza, zpracování), se do procesu zapojuje intuice. Řešitel získává od problému vědomě odstup, zpracovává získané informace. Díky

tomuto odstupu je analytické myšlení odsouváno do podvědomí, kde dochází k hledání řešení. Během toho procesu vznikají různé nekonkrétní varianty.

- *Iluminace* – během této fáze, která je rovněž intuitivní, se z řešitele stává objevitel, což znamená, že řešitel dospěje ke konkrétnímu nápadu, jak daný problém vyřešit.
- *Verifikace* – je poslední fáze procesu, při které dochází k ověření tohoto řešení vůči původní problémové situaci a zda ho lze k řešení použít.

Podle popisu by se mohlo zdát, že tento proces je časově náročný a tudíž pro výuku zcela nepoužitelný. Je však potřeba si uvědomit, že ve většině případech je čas řešení daného problému přímo úměrný složitosti problému.

Důležitou součástí tohoto procesu je analýza problému, která je základem jakéhokoliv řešení.

4. Srovnání informační výchovy na ZŠ

Informaticky zaměřený předmět je jedním z mála předmětů, které se průběžně mění podle vývoje nových technologií, a to jak díky vývoji výpočetní techniky, tak vývoji nových softwarových prostředí. Výuka by měla reagovat ve stejné míře na vývoj technologický i sociální. Bez výpočetní techniky si dnes již nedokážeme sociální interaktivitu vůbec představit, což společně s mnoha výhodami přináší také jistá rizika. Z tohoto důvodu je třeba žáky vychovávat i tímto směrem.

Z těchto důvodů proběhly v letech 2007 a 2013 rozsáhlé průzkumy, analyzující trendy výuky informatiky na základní škole. Těmito výzkumy se zabývají následující kapitoly, z nichž každá pojednává o konkrétním výzkumu.

Postavení informatiky jako vyučovacího předmětu a popisu hlavních tématických celků je věnována vždy jedna podkapitola u každého výzkumu.

4.1 Algoritmizace a programování na ZŠ v roce 2007

V letech 2006 a 2007 proběhl za podpory MŠMT rozsáhlý výzkumný projekt mapující stávající situaci informačně technologické výchovy na základních školách pod pracovním označením VIV06. (Rambousek et al., 2007)

Hlavním cílem tohoto výzkumu bylo poznat aktuální stav v oblasti vzdělání, které na rozdíl od jiných vzdělávacích oblastí vykazuje rychlý vývoj, kterému je třeba se neustále přizpůsobovat. Explorativní metodou tohoto výzkumu bylo zvoleno dotazníkové šetření, které bylo předloženo pedagogům informaticky zaměřených předmětů na základní škole.

Tato práce je zaměřena na dílčí výsledky tohoto dotazníkového šetření. Předem musí být zmíněn zásadní fakt tohoto šetření, na který v úvodu své práce upozorňují sami autoři.

Tím faktem je skutečnost, že navzdory značnému zkoumanému vzorku nebylo možné zajistit plnou reprezentativnost vzorku srovnatelnou s náhodným výběrem[1].

[1]Základní charakteristika výzkumu. [ZPRACOVAL KOLEKTIV ŘEŠITELŮ .. POD VEDENÍM VLADIMÍRA RAMBOUSKA]. *Výzkum informační výchovy na základních školách*. Plzeň: Koniáš, 2007, s. 215-238. ISBN 8086948102.

Z tohoto důvodu je při vyhodnocování výsledků dotazníkového šetření nutné pamatovat na skutečnost, že výsledky tohoto šetření lze vztahovat k danému vzorku respondentů, avšak k zobecňování těchto závěrů by mělo docházet velice opatrně.

4.2 Primární zaměření informační výchovy

Z výzkumu informační výchovy na základních školách realizované v letech 2006 až 2007 vyplývá několik závěrů.

Informační výchova na základních školách je realizována v rámci povinného a povinně volitelného informaticky zaměřeného předmětu, zájmových aktivit a ostatních vyučovacích předmětů. Z výzkumu dále vyplývá, že v zastoupení při rozvíjení informačně technologických kompetencí se největší měrou podílí neinformaticky zaměřené předměty (Rambousek et al., 2007).

„Polovina učitelů informatických předmětů deklaruje, že jejich informačně technologické kompetence jsou na úrovni, kterou sami ještě pokládají za minimální ještě přijatelnou pro kvalitní realizaci výuky informatických předmětů na základní škole. Třetina učitelů pak v tomto smyslu dokonce deklaruje, že jejich kompetence této minimálně akceptovatelné úrovně nedosahují.“ (Rambousek et al., 2007, s. 14)

Toto je asi jedním z důvodů závěrů tohoto výzkumu, který deklaruje, že obsah a pojetí informačního vzdělávání na ZŠ je do značné míry ovlivněno právě informačně technologickými kompetencemi pedagogů. Proto je výuka převážně zaměřena na uživatelské dovednosti ovládání počítače a tvorbu digitálních dokumentů, jelikož v této oblasti dosahuje převážná většina pedagogů potřebné úrovně. Z výuky informatiky jsou eliminována témata, která jsou považována za náročnější a také ta, ve kterých nemají pedagogové dostatečné znalosti. (Rambousek et al., 2007)

Jedním z takto dotčených tematických celků je právě oblast algoritmizace a programování, která byla většinou pedagogů řazena až na poslední místo v seznamu tematických oblastí informačních a komunikačních technologiích (dále jen jako ICT) Rámcově vzdělávacího programu pro základního vzdělávání (dále jen jako RVP ZV). Většina z nich tuto oblast považuje pro úroveň základního vzdělání příliš náročnou a odsouvají jí do úrovně středoškolského nebo vysokoškolského vzdělání (Rambousek et al., 2007, s. 102-103).

4.3 Algoritmizace a programování na ZŠ v roce 2013

V letech 2012 – 2013 proběhl výzkum *Informačně technologické kompetence dětí a jejich rozvoj na základních školách*, jehož hlavním cílem bylo zmapování aktuálního stavu a struktury výuky informačně technologických předmětů. Tento projekt volně navazuje na výzkum vedený v letech 2006 – 2007, jehož závěry směrem k oblasti algoritmizace byly shrnuty v předchozí kapitole.

Přestože je hodnocení zaměřeno na výsledky týkající se tematického celku *Algoritmizace a programování*, jsou zmíněny i souhrné výsledky mapující celkový stav vzdělávací oblasti.

4.3.1 Primární zaměření informační výchovy

V rámci výzkumu byla shromážděna data od 1183 učitelů informaticky zaměřených předmětů na základní škole z celkového počtu 3500 náhodně oslovených základních škol. (Černochová 2013, s. 7), přičemž genderové rozdělení pedagogů bylo přibližně rovnocenné.

Na rozdíl od předchozího výzkumu se posiluje pozice povinné informatiky, co by dominantní aktivity, ve které je rozvíjena informační gramotnost žáků. Nicméně ani ostatní neinformatické předměty, ve kterých se při výuce využívá informačních technologií, nejsou v tomto ohledu zanedbatelné.

Stávající koncepce oblasti ICT je v RVP ZV zastaralá a nereflektuje výrazný pokrok v této oblasti vzdělání. Chybí zde jasně daná koncepce výuky a vše je příliš obecné. (Černochová 2013)

Je požadována restrukturalizace hodinových dotací pro informaticky zaměřené předměty, případně začlenění příbuzných témat do učiva ostatních předmětů.

Výzkum také potvrdil závěry z roku 2006 ohledně náplně a obsahu informatických předmětů, který se v převážné většině základních škol redukuje na uživatelské dovednosti, které zahrnují práci se základními kancelářskými aplikacemi, práci s vyhledávačem a povědomí o právních a etických aspektech při práci s informacemi.

Nové šetření opětovně potvrzuje tendenci odsouvat algoritmizaci a programování až do středoškolského vzdělání.

V malé míře se zvýšil poměr pedagogů s alespoň minimální nebo vyšší informaticky technologickou kompetencí, která je přijatelná pro realizaci výuky. Pouze pětina dotázaných pedagogů označila své ICT kompetence za nedostatečné k realizaci výuky.

Také se ukazuje nový trend multiplatformní výuky, který reflektuje na současné možnosti, bohužel je užíván především při práci s kancelářskými aplikacemi.

4.4 Vývoj algoritmizace a programování na ZŠ do roku 2020

Jelikož je informatický obor jedním z nejvíce rozvíjejících se vědních oborů, musí na tento trend reagovat také školní informatika, jejíž postavení mezi povinnými předměty základního vzdělání se v průběhu času mění. V následující kapitole shrnu stávající stav informačně zaměřených předmětů a nastíním vizi MŠMT, jak by se měla výuka informatiky v budoucnu změnit.

4.4.1 Současný stav informační výchovy

Povinný předmět práce na počítači, informatika či alternativní název, který si škola zvolila v rámci svého školního vzdělávacího programu (dále jen ŠVP), a který musí reflektovat (rámcové) vzdělávací cíle dané v RVP ZV, se vyučuje jen 45 minut týdně, a to jeden rok na prvním a jeden rok na druhém stupni. Zařazení povinně volitelného předmětu s obdobným zaměřením je na uvážení vedení školy, nikoliv však povinnost. Během této nízké hodinové dotaci si žáci většinou osvojí pouze základní uživatelsky zaměřené dovednosti – tvorba textu, tabulky a prezentace. (Černochová 2013)

Přestože oblast Informační a komunikační technologie je zakotvena v RVP ZV pro základní školu, zaměření této oblasti se nezměnilo od roku 2004, což je jedním z nedostatků. V té době ještě nebyly komunikační technologie stejně vyspělé jako dnes, byla spuštěna sociální síť Facebook[2], dnes jeden z nejpoužívanějších internetových prohlížečů Google Chrome vznikl až o 4 roky později[3] a cloudové řešení[4] dokonce až o 8 let později.

[2]http://www.knowyourmobile.com/apps/facebook/21807/history-facebook-all-major-updates-changes-2004-2016
[3]https://www.root.cz/clanky/webove-prohlizece-soucasni-lidri-a-jejich-historie-1-2/
[4]https://cs.wikipedia.org/wiki/Disk_Google

Z výše zmiňovaného důvodu se RVP ZV zaměřilo pouze na schopnost vyhledávat informace na internetu a vytvářet dokumenty v běžně dostupných kancelářských balíčcích. Nic ale nebránilo tomu vyučovat informatiku směrem k tvůrčím přístupům a algoritmizaci. Přesto byla oblast zaměřená na algoritmizaci a algoritmické myšlení naprosto opomenuta, z čehož vyplývá skutečnost, že musel existovat ještě jiný a nejspíše zásadnější důvod, proč se nové tvůrčí přístupy ve výuce neuplatnily.

Tímto důvodem je absence aprobovaných pedagogů zaměřených na ICT oblast, jejichž počet se podle výzkumů blíží až 70 %. Druhým důvodem, který byl zmíněn a prokázán v rámci výzkumu Katedry informačních technologií a technické výchovy Pedagogické fakulty UK, je skutečnost, že většina dotázaných pedagogů považuje svoji erudovanost v oblasti informačních a komunikačních technologií za minimum, které je dostačující k výuce informatiky. Toto minimum je z větší části tvořeno uživatelskými dovednostmi použití základního balíku kancelářského softwaru. Někteří dotázaní pedagogové deklarovali, že nedosahují ani této minimální úrovně. (Černochová 2013)

4.5 Strategie digitálního vzdělávání do roku 2020

„Informační technologie by měly prostupovat celým procesem výuky na základních školách, nikoli jen v předmětech typu ‚Práce s počítačem'. Plné zapojení moderních technologií do výuky všech předmětů vnímá stát jako nezbytné v rámci posunu vzdělávacího systému od prostého memorování faktů k důrazu na čtenářskou gramotnost, komunikační dovednosti a logické myšlení."(STRATEGIE DIGITÁLNÍHO VZDĚLÁVÁNÍ DO ROKU 2020, s. 2).

Vize digitálního vzdělávání by se dala shrnout do následujících bodů.

Následující výčet je čerpán ze *STRATEGIE DIGITÁLNÍHO VZDĚLÁVÁNÍ DO ROKU 2020.*

Otevřené vzdělávání: je nový přístup k modernímu způsobu vzdělávání. Mělo by se jím stát celoživotní vzdělávání, které nebude závislé na konkrétním místě a tím lze příležitost nabídnout bez rozdílu každému jedinci. Systém otevřeného vzdělávání bude zajišťovat kvalitní přístup ke sdíleným zdrojům, jako např. levné a vysokorychlostní připojení nezávislé na lokaci nebo použité digitální technologii, či kvalitní vzdělávací obsah.

- **Digitální gramotnost:** jedná se o osvojení digitálních kompetencí, které sdružují vědomosti, dovednosti, postoje a hodnoty nezbytné pro tvořivé využívání digitálních technologií v široké oblasti lidského života.
- **Informatické myšlení:** je nový pojem, který popisuje způsob uvažování, který při řešení složitějších problému využívá informatické řešení, zahrnující schopnost analýzy, syntézy a zobecňování. Je zde rozvíjena schopnost přesné formulace problému.
- **Digitální technologie ve vzdělávání:** prostupují všemi oblastmi lidské činnosti, proto je nezbytné tento trend zohlednit i ve školním prostředí, kde se digitální technologie čím dál tím více zapojují do běžného vyučování, nejen informaticky zaměřených předmětů. Stávají se tak další pedagogickou a didaktickou pomůckou, díky které mohou učitelé ve svých hodinách nejen připravit různé formy netradiční výuky, ale také rozvíjet digitální gramotnost u svých žáků. Tento trend popisuje a také v budoucnu předvídá několik prestižních studií, z nichž k té nejprestižnější patří Horizont Report[5].

V závěru této kapitoly jsou zmíněny hlavní cíle Strategie digitálního vzdělávání do roku 2020.

- **Snižování nerovností ve vzdělávání –** díky otevřenému vzdělávacímu systému bude poskytnuto vzdělávání všem, kdo o něj projeví zájem, aniž dojde k omezování vzhledem k socioekonomickému statusu, regionu nebo kulturně

[5] NMC (2014). THE NMC HORIZON REPORT: 2014 K-12 EDITION. [online] http://bit.ly/ZvclAm

odlišnému prostředí. Aby ovšem mohl tento systém ve všech směrech splňovat své cíle, je nutné zajistit kvalitní přístup k těmto technologiím také mimo školu, což není možné v této době stoprocentně splnit.

- **Podpora kvalitní výuky a učitele –** z důvodů naplnění potřeb trhu práce, kdy stále více oborů je víc či méně závislých na informačních dovednostech, je potřeba přehodnotit současné vzdělávací cíle, aby lépe reflektovaly budoucí požadavky na zaměstnance. Jedním z těchto cílů je celoživotní vzdělávání, které vychází ze stále častější potřeby rekvalifikace pracovních sil v důsledku mnohem kratší doby strávené v konkrétním zaměstnání. Aby se mohl tento trend plně zapojit do výuky, je třeba zvýšit pedagogické dovednosti v oblasti digitálních technologií. Jde především o informatické myšlení, které mohou poté předávat svým žákům. Také jde o dovednosti uživatelské, potřebné např. pro tvorbu digitálních materiálů, což řada pedagogů prokázala tvorbou rozsáhlé databáze digitálních učebních materiálů v rámci projektu *EU peníze školám.*

5. Projekt PRIM

Zdrojem této kapitoly je http://www.imysleni.cz.

Strategický tříletý [6] projekt PRIM vznikl díky poptávce po nové orientaci informatiky, směřující od uživatelského ovládání ICT směrem k základům informatiky.

Garantem projektu je Pedagogická fakulta Jihočeské univerzity v Českých Budějovicích. Participují na něm všechny pedagogické fakulty v ČR a Národní ústav ve vzdělávání.

Tento projekt je primárně zaměřen na podporu práce učitele.

Projekt PRIM má za cíl rozvoj informatického myšlení v předmětu informatika na všech stupních škol od mateřských po střední.

Hlavními úkoly jsou:

- Tvorba školního kurikula
- Tvorba vzdělávacích materiálů pro školy
- Školení pro učitele
- Inovaci výuky učitelů na pedagogických fakultách
- Masivní e-learningové kurzy pro veřejnost
- Podpora inovací v oblasti školské informatiky
- Popularizace informatiky a autorského přístupu k počítači.

[6] Projekt probíhá od října 2017 do září 2020

6. Programovací jazyky

Při rozhovoru dvou lidí je základním verbálním komunikačním nástrojem společný jazyk, díky němuž se tito dva lidé mezi sebou dorozumí. Stejná znalost komunikačního nástroje je nutná v případě „rozhovoru“ uživatele s počítačem.

Komunikačním nástrojem se v tomto případě stává tzv. programovací jazyk, který je v následujících kapitolách členěn do různých skupin a podskupin.

Existuje mnoho způsobů jak definovat programovací jazyk.

Je to umělý jazyk určený pro zápis počítačového programu, ve kterém jsou přesně definované syntaxe a sémantika (Vitovský 2006).

Jednotlivé programovací jazyky se od sebe liší svou syntaxí a sémantikou, které stejně jako v případě cizích jazyků mohou mít velmi podobnou formu.

Müller (2002) charakterizuje syntaxi programovacího jazyka jako souhrn pravidel udávající přípustné tvary dílčích konstrukcí a celého programu. Sémantika v obecném významu pojednává o významu slov a znaků. Nejinak je tomu v případě programování, kde je sémantikou přiřazen význam dílčích programových konstrukcí, které splňují syntaktické požadavky.

Programovací jazyky jsou rozdělovány podle různých kritérií. Tato práce je zaměřena hlavně na propedeutické programovací jazyky, které jsou nejvhodnější pro počáteční seznámení s programováním.

Kapitola je rozdělená do dvou hlavních kategorií, kterými jsou propedeutické programovací jazyky a klasické programovací jazyky. Vzhledem z primárnímu zaměření programování na děti, bude rozvinuta převážně první skupina programovacích jazyků.

6.1 Propedeutické programovací jazyky

Pokud člověk k programování používá programovací jazyk, téměř vždy narazí na stejné úskalí. Než vytvoří svůj první program, byť pouze s textovým výstupem, musí napsat několik řádků programového kódu. Zápisu tohoto programu ovšem předchází zvládnutí alespoň základní syntaxe a příslušné sémantiky, bez kterých by nebylo možné vytvořit ani ten nejjednodušší program.

Tuto nesnáz, která může stát za odrazením od programování jako takového, odstraňují propedeutické programovací jazyky.

Na začátek je nutné vysvětlit, co je to propedeutika.

Průprava. Všeobecná příprava ke studiu určitého oboru, často ve formě úvodního kurzu na vysokých školách apod. (Průcha 2013, s. 185)

Slovo propedeutika lze rozdělit na dvě části. První částí je slovo *pro*, které lze v řečtině přeložit slovem *před,* druhou část představuje slovo *pedeutika*, jež vychází z řeckého slova *paideúein* a lze ho přeložit jako *vyučování*. Z toho lze odvodit, že slovo *propedeutika* označuje předběžnou výuku či úvod do určité vědy. Vzhledem k tomu, že předmětem této propedeutiky je programování, jedná se o programovací jazyk, který příznivě rozvíjí předpoklady pro následné studium dalších programovacích jazyků. (Propedeutika 2017)

Jak již bylo zmíněno v prvním odstavci této kapitoly, volba klasického programování je spojena se zápisem programového kódu.

Při tvorbě programového kódu mohou žáci využívat různé editory podporující konkrétní programovací jazyk. Přes veškeré usnadnění, spočívající v dokončování názvů příkazů, hlídání syntaxe a zvýrazňování různých částí programového kódu, jim neodpadá znalost jednotlivých příkazů nebo alespoň jejich začátků.

Tradičně prvním programem se v učebnicích programování stala věta „Hello World“ zobrazená na počítačovém monitoru. Tento styl programování není pro malé děti seznamující se s problematikou algoritmizace a programování moc vhodný, jelikož výstup programu je pro ně málo atraktivní.

Předností propedeutických programovacích jazyků je především velmi rychlá odezva na programový kód, která je na rozdíl klasických jazyků v grafické podobě, což je pro děti a žáky mnohem atraktivnější než čistě textový výstup.

Neméně důležitou skutečností je lokalizace většiny propedeutických jazyků do českého jazyka, čímž odpadá problematika s překládáním. Díky tomu jsou tyto jazyky přístupné široké skupině dětí různého věku, kdy jediným omezením je schopnost ovládání daného prostředí a orientace v něm.

Propedeutické programovací jazyky jsou primárně zaměřeny na rozvoj algoritmického myšlení. Žáci se nemusí zabývat způsobem zápisu, ale místo toho se mohou cele soustředit na jednotlivé části programu a jejich logickou návaznost.

Klíčový proces, nutný pro úspěšné vytváření počítačového programu, je analýza. Následuje řešení dílčích problémů na základě různých programových struktur, tedy algoritmické myšlení. Konkrétní typy propedeutických jazyků budou zmíněny v následujících podkapitolách.

Jejich cílem není vychovat z každého žáka programátora, ale naučit nejlépe všechny žáky algoritmicky přemýšlet. Proto by se tyto jazyky měly primárně využívat na základní škole, případně v předškolní výuce.

Členění propedeutických programovacích jazyků je závislé na míře využití informačních technologií. K výuce algoritmizace není počítač zcela nezbytný, jak bude popsáno v další části bakalářské práce.

6.1.1 Programování bez počítače

Zdrojem této kapitoly je https://code.org/curriculum/unplugged.

Kapitola pojednává o počátcích programování, které jsou charakteristické svou nezávislostí na výpočetní technice a počítačovém prostředí. Jedná se o specifický přístup k rozvoji algoritmického myšlení. Tuto metodu lze tudíž využít také při mimoškolních aktivitách v rozličných kolektivních hrách.

Při používání této metody nejste odkázáni na použití počítačů, dá se využívat a rozvíjet i mimo typické informaticky zaměřené předměty. Velmi vhodné využití je v hodinách tělocviku, výtvarné výchovy a samozřejmě v hodinách matematiky. Metody lze využít převážně na 1. stupni ZŠ a v mimoškolních aktivitách.

Programování bez počítače (Unplugged metody) se rozdělují do čtyř kategorií dle objektu programování:

- programování spolužáka
- programování pomocí symbolů
- řazení známého postupu
- programování robota.

Všechny výše zmíněné kategorie budou konkrétně popsány v následujících kapitolách.

6.1.1.1 Živý robot

Jednou z prvních aktivit rozvíjející algoritmické myšlení, kterou lze vykonávat kdekoliv a není třeba zvláštních příprav, je hra na živého robota. Byla vymyšlena autorem práce v rámci zájmového kroužku Programování na Základní škole Karla IV. v Ústí nad Labem. Aktivita spočívá v ovládání jiného spolužáka pomocí předem dohodnutých příkazů. Tyto příkazy mohou být verbální nebo nonverbální. V případě, že k vedení spolužáka využíváte slovní příkazy, můžete vedenému zavázat oči, aby nemohl ovlivňovat průběh procesu. Poté stačí vytyčit určitou trasu. Pokud je tato aktivita prováděna ve třídě, je možné vytvořit bludiště z lavic. V případě praktikování hry na živého robota v přírodě je možné zapojit rozličné přírodní překážky a záleží jen na fantazii tvůrce.

Úkolem „programátora“ je vymyslet příslušný sled příkazů, díky nimž provede svého spolužáka předem připraveným bludištěm, aniž by do něčeho narazil.

Tento program je vhodné v počáteční fázi tvořit na základě vlastní zkušenosti, při které si programátor projde příslušnou trasu sám a přitom si zaznamenává sled pohybů, které během své cesty vykonal. Pomocí takto vytvořeného sledu událostí poté může ovládat svého spolužáka, aniž by došlo k jeho kolizi s nějakou překážkou na trati.

V druhé fázi se programátor pokusí tento sled příkazů vytvořit pouze na základě tvaru a obtížnosti trati, aniž by si jí sám vyzkoušel. Tím procesem začíná vytvářet svůj skutečný program, jelikož si musí předem promyslet jednotlivé úkony na základě aktuální virtuální pozice.

Přesnější představa této metody je popsána v kapitole č. 15.2 - Lidský automat.

6.1.1.2 Grafické programování

Obdobou první aktivity je grafické programování, při kterém je využit předem připravený plán s určitou situací a sada příkazů, které mohou žáci ke své činnosti využívat.

Jejich úkolem je vytvořit sekvenci příkazů ovládající virtuální objekt provádějící předem připravené zadání úlohy.

Pomocí této metody si mohou žáci rozvíjet lineární programování, kdy celý program zapisují formou jedné řady příkazů, případně se mohou pokusit pro opakující se úkony vytvořit sekvenci příkazů, kterou posléze označí za funkci či proceduru, díky níž lze zápis celého programu tvořit mnohem efektivněji a přehledněji.

Konkrétní aplikace této metody je popsána v kapitole č. 15.1 – Grafické porgramování na papíře.

6.1.1.3 Algoritmy běžného života pomocí kartičkové metody

Kartičková metoda je jedním z přístupů rozvíjení algoritmického myšlení. Hlavním úkolem žáků je vymyslet správný sled předem připravených událostí vedoucích k zadanému cíli.

Takovým cílem může být pracovní postup, který je všeobecně znám, například popis všech činností konajících při cestě do školy. Mnohem náročnější bude pro žáky tyto aktivity definovat a popsat, než z předem připravených situací zvolit správné pořadí těchto aktivit. V případě prvního úkolu (definice a popis činností) je nutná pomoc pedagoga.

K výuce algoritmizace a programování lze přistupovat mnoha způsoby. K výuce různých algoritmických postupů a k jejich osvojení je možné zvolit několik rúzných metod. Tato kapitola se zabývá srovnáním jednotlivých metod a představením zástupce, kterého lze v této metodě využít.

6.1.2 Programování s počítačem

Nezbytným předpokladem pro práci s programovacími jazyky tohoto typu je schopnost číst. Programovací kód není nutné zapisovat do programovacího prostředí, jelikož je v mnoha případech textový příkaz nahrazen příslušnou ikonou nebo grafickým blokem kódu.

Jednotlivá specifika těchto typů propedeutických programovacích jazyků jsou podrobněji rozepsány v následujících podkapitolách, které jsou zaměřené jak na konkrétní představitele jednotlivých algoritmických přístupů, tak na jejich vhodnost pro různé věkové kategorie a dostupnost ve školním prostředí.

6.1.2.1 Textově orientované propedeutické programovací jazyky

Textově orientované propedeutické programovací jazyky jsou jedním ze základních typů programovacích jazyků, které jsou založeny na zápisu samotného programového kódu, což sebou nese jisté úskalí.

Úskalím je syntaxe daného kódu. Žák musí znát nejen příslušný příkaz kódu, ale také se předpokládá základní znalost slovní zásoby daného programovacího jazyku a znalost parametrů, které jsou pro správnou funkci vyžadovány. Pro zjednodušení práce v daném programovacím prostředí existuje velmi často výčet přípustných příkazů společně s vysvětlením funkce, čímž odpadá nutnost se tyto příkazy učit zpaměti, jak je to velmi často nutné u klasického programování.

Názvy jednotlivých příkazů jsou intuitivní, vycházejí z výsledného efektu daného příkazu, proto si je žák již při krátkodobém používání velmi rychle osvojí.

Při programování je nezbytné veškeré příkazy zapisovat ručně, což není zcela vhodné pro děti nižších věkových kategorií. Výhodou tohoto zápisu je možnost libovolného rozšíření portfolia příkazů na základě stávajících, čímž je možné si daný jazyk upravit dle individuálních požadavků. Tvorbou nových příkazů se žáci naučí efektivnější zápis opakujícího se programového kódu, díky nimž se program stává přehlednějším. Žáci si osvojí dovednost rozboru komplexní úlohy na menší samostatné celky, což je základní myšlenkou algoritmizace. Tuto dovednost mohou žáci později využít při tvorbě procedur a funkcí u klasických programovacích jazyků.

Základní programovací konstrukcí těchto jazyků je cyklus definovaný počtem opakování, který může sdružovat frontu více příkazů tvořících logický celek. Další konstrukcí je tzv. *rekurze*, kdy samotný příkaz je volán z těla daného příkazu.

Hlavními představiteli je programovací jazyk Logo (viz kap. 9.1), Karel (viz kap. 9.2) a Petr.

6.1.2.2 Stavebnicově orientované propedeutické programovací jazyky

Druhou skupinou jsou stavebnicově orientované jazyky, jejichž hlavní předností je absence zápisu jednotlivých příkazů a programovacích struktur. Díky této koncepci odpadá problém s nesprávnou syntaxí, kterou je nutnou dodržovat u předchozí skupiny programovacích jazyků. Tento typ programovacího jazyku se vyznačuje typickým

uspořádáním připomínající skládání dílků puzzle. Díky tomuto rozvržení je do jisté míry eliminována možnost vzniku zásadní chyby, jelikož lze k sobě spojovat pouze prvky určitého typu, které spolu mají logickou souvislost a návaznost. Také jsou zcela eliminovány chyby syntaxe, jelikož žák nemusí příslušný příkaz zapisovat, ale vybírá si z připravených „příkazových“ modulů, na kterých je daná instrukce nebo programová konstrukce formálně zapsána, čímž se odlišuje od ikonických jazyků, kde se každý příkaz liší rozdílným vzhledem ikony. Jedním z těchto jazyků, je programovací nástroj Scratch, který bude ještě podrobněji zmíněn.

Zásadním rozdílem mezi stavebnicovým a kartičkovým programovacím jazykem je skladba programu, která se odlišuje možností libovolného umístění jakékoliv z připravených karet do programového kódu, čímž není možné zajistit stejnou kontrolu jako v předchozím příkladu. To dává žákům větší volnost při vytváření programu a více prostoru pro tvorbu chyb, které následně musí odstraňovat, a tím také více přemýšlet o celém programu i jeho částech.

Nevýhodou těchto jazyků je pouze omezená lokalizace, kde dominantní úlohu hraje bezesporu angličtina. Tím se toto prostředí, které je svým vzhledem opět cíleno na mladší generaci, stává nevhodným pro základní školu, jelikož je zde vyžadována znalost anglického jazyka. Možná i z tohoto důvodu nejsou u nás tyto jazyky tolik rozšířené a známé.

Mezi typického představitele patří programovací prostředí Etoys (viz kap. 9.4).

6.1.2.3 Ikonické propedeutické programovací jazyky

Ikonické programovací jazyky sázejí na jednodušší tvorbu programového kódu, kdy se žák může plně soustředit na algoritmickou část úlohy, aniž by se coby „programátor“ zatěžoval syntaktickou stránkou daného problému.

Programový kód je vytvářen v programovacím prostředí grafického charakteru, které je možno ovládat pouze myší, případě herním ovladačem, čímž jej lze provozovat také na zařízeních s absencí klávesnice. Žák si vybírá z nabídky ikon zastupujících různé funkce a příkazy, a poté je skládá do fronty příkazů, čímž vytváří program. Typickým představitelem tohoto typu jazyku je programovací jazyk Baltík od firmy SGP Systém, který se stal jedním z nejrozšířenějších jazyků na základních školách (viz kap. 9.5).

6.1.2.4 Výukové herní prostředí

Podle všeobecně známého hesla Jana Ámose Komenského „Škola hrou“ je pro žáky naprosto ideální a navíc nejprogresivnější taková metoda výuky, při které studenti vůbec nepostřehnou, že k edukaci dochází. Při výuce je využíváno známé prostředí, které mají žáci spojené s hraním počítačových her. Herních prostředí tohoto typu je jistě celá řada, např. rozličné simulátory učící děti ovládat nějaký stroj či nástroj nebo simulující určité fyzikální, chemické a matematické prostředí. Pro potřebu bakalářské práce se však zaměříme pouze na simulátory sloužící pro výuku programování.

V této oblasti existuje celá řada možností, lišících se převážně měrou zastoupení samotného programovacího kódu. Všechny tyto programy jsou však založeny na velmi podobném principu, a to pomocí souboru příkazů a cyklů ovládat pohybující se postavu či tvořit různé objekty a předměty.

Nejznámějším prostředím tohoto typu je Minecraft od společnosti Microsoft, který propojuje herní a výukovou aktivitu (viz kap. 11.1).

6.2 Klasické programovací jazyky

V této kapitole bude stručně pojednáno o klasických programovacích jazycích. Klasické programovací jazyky sice nejsou předmětem této práce, ale jsou nedílnou součástí programovacích dovedností a je zde také integrace zápisu syntaxe do mnoha propedeutických jazyků.

Oproti jazykům rozebíraným v předchozí kapitole nabízejí klasické programovací jazyky mnohem více možností, které souvisejí převážně s jejich zaměřením. Hlavním rozdílem je možnost využití složitějších datových struktur a algoritmických konstrukcí, které jsou při tvorbě náročnějšího programu nezbytné pro jeho funkci. Jelikož je však tato tvorba spojena se zápisem těchto konstrukcí v příslušném programovacím jazyku, je jejich zařazení vhodnější k výuce až na střední škole.

Důvodů pro zařazení klasických programovacích jazyků na střední, případně vysoké školy, je několik. Prvním z nich je znalost syntaxe příslušného programovacího jazyka a druhým je určitá neatraktivnost, který spočívá v textovém nebo nenáročném grafickém výstupu.

6.2.1 Propojení klasických programovacích jazyků s jazyky propedeutickými

Tato kapitola se věnuje využití propedeutických programovacích jazyků jako nástroje k usnadnění přechodu ke klasickým programovacícm jazykům.

Jak bylo již zmíněno, propedeutické jazyky slouží primárně k osvojení základních programovacích dovedností, kam je v prvé řadě řazeno algoritmické myšlení a osvojení základních programovacích konstrukcí, které žáci později využívají při výuce klasického programovacího jazyka. Zde je dobré zdůraznit, že správná volba prvotního jazyka má velký vliv na pochopení následujících programovacích struktur.

Vzhledem k velkému důrazu na objektově orientovaný přístup k programování je výhodné volit takové zástupce propedeutických jazyků, které jsou ve své podstatě založeny na stejném přístupu. Typickými zástupci jsou jazyky, které jsou využívány v programovém prostředí KODU Game Lab a Scratch. Tyto programy jsou však závislé na výpočetní technice, která v nižších ročnících základní školy nebo předškolní výuce nemusí být vždy dostupná, respektive dostupná v dostatečném počtu. Pro toto období je velmi vhodné využívat programovací roboty, jejichž ovládání a řízení probíhá přímo v přístroji, případně skrze čidla, reagující na změnu prostředí. Při této práci si žáci nejvíce osvojí tvorbu systematických kroků vedoucích k zadanému cíli.

Jiné jazyky mohou zobrazovat vytvořený programový kód, pomocí ikonického jazyka nebo soudobého programovacího jazyka, typicky C#. Při použití těchto jazyků je poté mnohem snadnější cesta k přechodu k samotnému C# nebo jiným programovacím jazykům.

Ať už si pro začátky ve výuce programování a algoritmizace zvolíme jakýkoliv typ, nesmíme zapomínat na skutečnost, že každý z uvedených jazyků má své typické zaměření, z čehož plyne, že nelze od těchto jazyků požadovat stejné možnosti, které jsou později v klasickém programování běžné a více méně nezbytné.

7. Srovnání propedeutických programovacích jazyků

Tato kapitola je zaměřena na vzájemné srovnání propedeutických programovacích jazyků. Většina licencovaných produktů má vytvořen velmi dobrý multilicenční systém zaměřený právě na oblast vzdělávání, tudíž nic nebrání jejich využití při hodinách ve škole.

7.1 Věkové srovnání jednotlivých metod

Při výběru vhodné metody je třeba mít na paměti několik důležitých kritérií. Je třeba přizpůsobit metodu věkovému rozmezí žáků. Pro předškoláky a mladší žáky je nutná atraktivita rozvíjející motivaci a algoritmické myšlení. Dalším hlediskem je množství osvojených příkazů potřebných k tvorbě vbyraného programu.

Pokud vybíráme metody pro žáky, kteří mají výše zmiňované základy osvojené či budou při své práci potřebovat využít pokročilejších funkcí klasických programovacích jazyků, je nutné se zaměřit na funkcionální možnosti dané metody.

7.2 Funkcionální využití propedeutických programovacích jazyků

Při volbě vhodného programového prostředí je třeba vzít v potaz různá funkcionální omezení, která jsou závislá na využití daného programu.

Toto omezení souvisí především s variabilitou a složitostí programového kódu. Některá programová prostředí jsou úzce zaměřena na konkrétní činnost. Lze zde vytvářet pouze programy obdobného charakteru, které jsou založeny na stejném principu, a není jiný způsob, jak zadaný problém řešit. Mezi takováto omezení patří absence tvorby vlastních proměnných, což je zcela běžné a hojně využívané v algoritmizaci. Na druhou stranu toto omezení pociťuje pouze uživatel, který má již jisté zkušenosti s jinými programovacími jazyky bez tohoto omezení.
Žák, který se poprvé setkává s programováním, toto omezení vůbec nepocítí, je však na pedagogovi vyučujícímu programování, aby volil taková zadání, ve kterých nebudou tyto nedostatky znát.

Příkladem tohoto prostředí je například KODU Game Lab (viz kap. 9.6).

8. Robotické hračky a programovatelní roboti

Tato kapitola je věnována novému trendu výuky algoritmizace, která je pro žáky mnohem atraktivnější a zábavnější. Jedná se o robotické hračky a stavebnice s možností změny jeho chování skrze vlastní programový kód. Díky těmto technologiím se žákům dostává okamžitá odezva na jimi vytvořený program, což zvyšuje jejich motivaci.

V následujících kapitolách budou zmíněni nejrozšířenější zástupci, kteří se buď používali v minulosti, nebo jsou dostupní na současném trhu.

Programovatelní roboti jsou ideální pro využití v začátcích programování, jelikož žáci své vytvořené programy rovnou převádějí do mechanické hračky, která vykonává jimi vytvořenou sekvenci příkazů, nazývající se program. Tyto robotické hračky nebo stavebnice jsou díky této funkci pro žáky velmi atraktivní, je však třeba dát velký pozor na to, zda jejich nadšení plyne z novosti této pomůcky nebo zda za tímto nadšením stojí jiný přístup při tvorbě programu. Obrovskou nevýhodou, převážně ve školním prostředí, je jejich finanční náročnost. Díky velkým nákladům na pořízení těchto stavebnic, které je více než vhodné zakoupit v několika exemplářích, aby bylo zaručeno optimální složení „vývojářského“ týmu, je pořízení těchto pomůcek ve většině případů úzce spjato s využitím některé z dotačních výzev.

9. Vybraní zástupci propedeutických programovacích jazyků

Vzhledem k zaměření práce budou podrobněji rozebrány propedeutické jazyky, které jsou vhodnější než právě klasické programovací jazyky. Předchozí kapitoly pojednávaly o hlavních rozdílech propedeutických programovacích jazyků, popisovaly klady a zápory jednotlivých typů a metod. Přestože se jednalo o široký výčet jednotlivých jazyků a metod, byl tento soubor pouze teoretický a pro laiky neposkytoval ucelenou představu.

Z tohoto důvodu představují následující kapitoly praktickou ukázku vybraných typů jazyků. Výběr těchto jazyků probíhal na základě oblíbenosti, četnosti či historického významu, na jejichž základě později vznikly další varianty a modifikace tohoto programovacího jazyka. Z důvodu ucelenosti mohou některé skupiny jazyků obsahovat více než jednoho představitele.

9.1 Logo a jeho varianty

Jazyk Logo je jednoduchý funkcionální programovací jazyk, který od svého vzniku v roce 1967 prošel mnoha obměnami [7] a k letošnímu roku existuje 197 implementací tohoto jazyka[8].

Hlavním představitelem tohoto jazyka je želva, která svým pohybem po pracovní ploše, či papíru, může zakreslovat trajektorii tohoto pohybu. Tímto způsobem vytváříme tzv. *želví grafiku*.

Myšlenka Loga je velmi podobná skutečné želvě pohybující se po písku, která svým ocáskem kreslí obrazec. Původně jazyk Logo ovládal skutečného robota, v pozdějších letech se celá koncepce převedla jen do virtuálního prostředí.

Pro účely školního prostředí jsou nejzajímavější dvě varianty, které jsou navíc přeloženy do českého jazyka. První implementací je Imagine Logo, které je zatíženo

[7]Logo (programovací jazyk). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-07-07]. Dostupné z: https://cs.wikipedia.org/wiki/Logo_(programovac%C3%AD_jazyk)

[8]LOGO. In: *Kids, Code, and Computer Science: Helps kids code + explore computer science* [online]. 2014 [cit. 2017-07-07]. Dostupné z: https://www.kidscodecs.com/logo-programming-language/

licenční smlouvou,[9] a proto zařazení tohoto programu do výuky musí být dobře zváženo. Přestože se jedná o kompletně objektově orientovaný jazyk, jehož koncepce je základní myšlenkou všech moderních programovacích jazyků, lze pro první seznámení s programováním využít neobjektového přístupu. Výhodou je možnost přechodu na objektově orientované programování v pozdějším věku.

Imagine Logo je nepřímým nástupcem druhé instance orientované na školní prostředí, kterou je Comenius Logo. Bohužel tato varianta je již zastaralá, a přestože byla vydána ve volně šiřitelné licenci, pro dnešní moderní operační systémy je nepoužitelná. Navzdory těmto faktům lze najít celou řadu programovacích prostředí, která jsou na tomto programovacím jazyku postavena. Tímto prostředím může být např. EasyLogo[10], což je volně šiřitelná desktopová aplikace, která žáky naučí základnímu programátorskému myšlení na základě tvorby cyklů a rekurze.

9.2 Karel

Stejně jako programovací prostředí jazyka Logo, byl v Čechách v 80. letech 20. století nejrozšířenějším programovacím prostředím jazyk Karel. Přestože se jedná o starší programovací prostředí, v sekvenčním programováním hraje nezastupitelnou roli. Hlavní metodou programovacího jazyka Karel je tzv. rekurze, která je velmi rozšířena v klasických programovacích jazycích.

Programovací jazyk byl vyvinut na přelomu 70. a 80. let 20. století učitelem programování ze Stanfordské univerzity Richardem E. Pattisonem[11], který ho pojmenoval na počest Karla Čapka, autora divadelní hry RUR – Rossums Universal Robots. V původním podání, které se k nám dostalo díky zásluze Doc. Jozefa Hvoreckého, Csc., měl program mnohem více příkazů, než v jakém podání ho známe v Čechách. Díky tomuto zjednodušení, o které se největší měrou přičinili Ing. Tomáš Bartovský, Csc. a Ing. Rudolf Pecinovský, Csc., se jazyk Karel mohl více přiblížit dětem. Oproti původnímu návrhu byla česká verze rozšířena o tzv. rekurzi[12].

[9]Jak získat Imagine. *Imagine* [online]. 2002 [cit. 2017-07-07]. Dostupné z: http://imagine.input.sk/cz/tutorial.html
[10]http://edi.fmph.uniba.sk/~salanci/EasyLogo/index.html
[11] Autor prvního programu a knížky „Karel The Robot“
[12] Rekurze je programovací technika, při níž je určitá procedura nebo funkce znovu volána dříve, než je dokončeno její předchozí volání.
(https://cs.wikipedia.org/wiki/Rekurze_(programov%C3%A1n%C3%AD))

Karel je robot, který se díky základní sadě příkazů, kterou lze rozšířit o vlastní programové sekvence, pohybuje po obdelníkové šachovnicové síti nazvané dvorkem pouze ve svislém a vodorovném směru. Kromě pohybu po těchto polích může robot pokládat či zvedat tzv. značky.

Alternativní variantou je objektivně orientovaná verze označená názvem Karel++ (po vzoru C++). Modifikací programovacího jazyka Karel++, založeného na programovacím jazyku C# je Karel H.

9.3 Scratch

Zdojem této kapitoly je https://scratch.mit.edu/.

Scratch je navržen pro cílovou skupinu dětí od 9 do16 let, čímž se nabízí jeho zařazení mezi vzdělávací pomůcku pomáhající v rozvoji algoritmického myšlení a programování na základní škole. Tomuto zařazení napomáhá nejen volná licence, ale také online varianta, díky níž je možné ho využít kdekoliv. Díky tomu se programovací jazyk Scratch vyřazuje ze skupiny edukačních metod, nepočítáme-li úzce zaměřené systémy na vývoj počítačových her.

Jedna z prvních vět, kterou si na stránkách autorů můžete přečíst, shrnuje veškeré náležitosti vhodné výukové metody.

Scratch pomáhá mladým naučit se myslet tvořivě, přemýšlet systematicky a spolupracovat — podstatné dovednosti pro život v 21. století. [13]

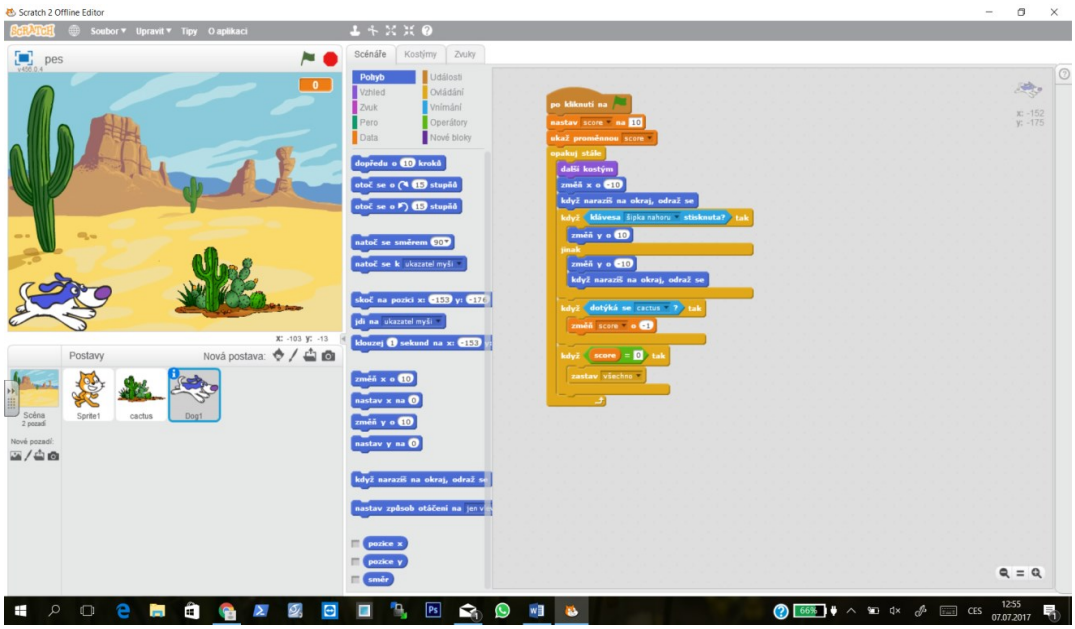
Toto tvrzení podtrhuje hned několik faktů, především lokalizace do více než padesáti světových jazyků a také propracovaný systém tutoriálů a rozsáhlé nápovědy pro jednotlivé oblasti či konkrétní příkazy. Tutoriály mohou žáci využít při hledání postupů konkrétních algoritmů, aniž by museli čekat na pomoc učitele. Tím se z tohoto programovacího nástroje stává nejen vynikající pomůcka pro samostudium, ale také výborná didaktická pomůcka badatelsky orientovaného učení, které rozvíjí samostatné myšlení a učí řešit problémy. (Dostál 2015)

Jak ve své práci uvádí L. Samková (2011, s. 336–341) badatelský proces lze chápat jako „činnost, při které pozorujeme, dedukujeme, nabízíme hypotézy, snažíme se

[13] https://scratch.mit.edu/about/

je ověřit, nemusíme dojít k žádnému konečnému závěru, závěry závisí na našem momentálním rozhledu a různí badatelé mohou interpretovat stejná fakta různě. Poslední tři znaky bádání v sobě skrývají onen most mezi teorií a praxí, mezi učebnicí a každodenní realitou. Jsou klíčem ke správnému chápání světa kolem nás.“

Celé programovací prostředí, které se opět přesouvá pouze do 2D prostoru, je rozděleno na oblast náhledu, zdrojového kódu a nabídky jednotlivých příkazů a konstrukcí, které jsou logicky členěny do jednotlivých kategorií, což usnadňuje jejich hledání. V tomto prostředí lze vytvářet prakticky veškeré programovací konstrukce, se kterými se žáci mohou setkat při programování pomocí libovolného klasického programovacího jazyka, což je bezesporu jeho velká přednost. Dále je možné při tvorbě programu (hry) využít rozsáhlé galerie grafických objektů (většina z nich disponuje několika variantami téhož typu, což napomáhá snadnější animaci objektu), případně lze využít integrovaného grafického editoru nebo již vytvořených grafických předloh. Tato výhoda je jednou z hlavních nedostatků programovacího jazyka KODU, který tuto možnost nenabízí a žáci si musí vystačit s vcelku omezeným výběrem dostupných objektů.



Obrázek 1 - Ukázka programového prostředí SCRATCH

Hlavním znakem této skupiny programovacích jazyků je využití předem připravených částí, ze kterých poté žáci skládají daný program.

9.4 Etoys

Etoys lze zařadit mezi programová prostředí kartičkového typu. Toto prostředí lze rozdělit na dvě hlavní části, z níž jedna je jednoduchý grafický editor, ve kterém si žáci připraví potřebné rastrové grafické předlohy, a druhou je část programová. Programová část je méně intuitivní než v případě programovacího jazyka Scratch, navíc bez znalosti anglického jazyka je dosti nepřehledná a obtížná. Programová část se skládá z kompletní nabídky jednotlivých příkazů, které si musí žák poskládat dle svého vytvořeného algoritmu.

Tento programovací jazyk byl do této práce zařazen zejména kvůli úplnosti přehledu, přestože nelze předpokládat hromadnější zapojení do výuky, vzhledem k výše zmíněným důvodům. Navíc je toto programovací prostředí po vizuální stránce velmi zaostalé, což nepřispívá k potřebné atraktivitě.

9.5 Baltík a jeho varianty

Hlavním informačním zdrojem pro tuto kapitolu byla dlouholetá osobní zkušenost autora, s tímto programovacím prostředím.

Mezi základní programovací jazyky patří programovací jazyk Baltík od společnosti SGP Systems, který je v České republice velmi rozšířen.

Přestože programovací jazyk Baltík patří mezi starší jazyky, je stále v mnoha základních školách velmi oblíben. První programové protředí s označením Baltazar vzniklo v roce 1993. Výhodou tohoto systému je možnost přechodu do prostředí klasického programování, který nabízí programovací jazyk C, potažmo C# v případě *Baltie C# 3D*.

Celý systém je založen na 2D grafickém prostředí, které v dnešní době může působit lehce primitivním a neatraktivním dojmem, což je příčinou obtížnější motivace žáků.

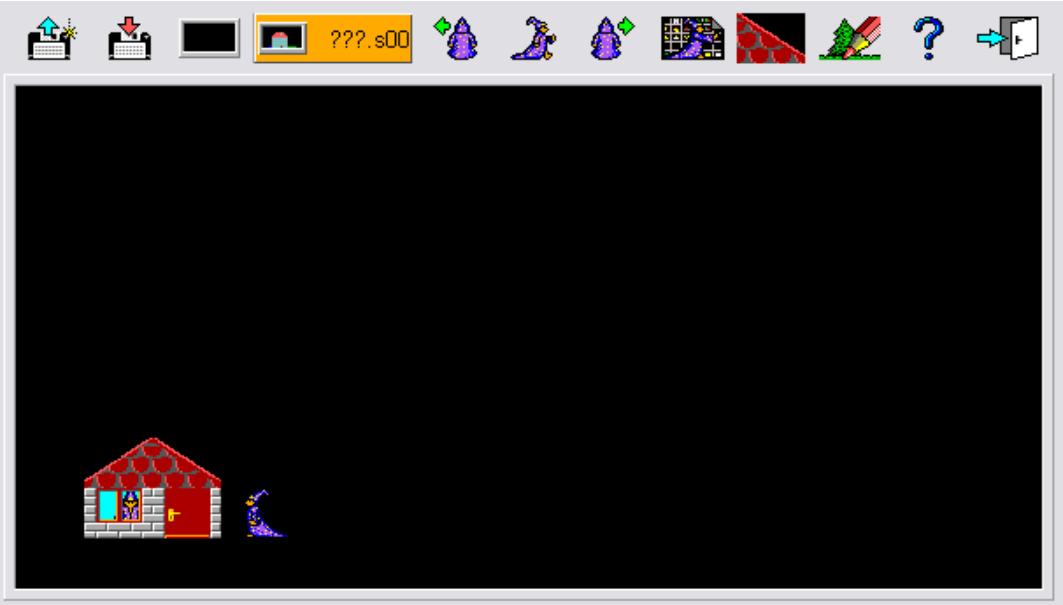
Práce je zaměřena na popis 2D verze Baltík 3.0, ve kterém autor vyučoval několik let programování v rámci zájmového kroužku.

Celý systém je rozdělen do 3 úrovní obtížnosti[14]

[14]https://www.sgpsys.com/cz/product_B3.asp

Skládej scénu – zde se mladší děti naučí pomocí počítačové myši skládat z připravených dílků (ikon) větší obrázky, které se v Baltíkovi nazývají scéna.

Čaruj scénu – je první úroveň programového prostředí, ve kterém se žáci učí čaroděje ovládat. Tato dovednost je zatím omezena pouze na pohyb v ploše a čarování příslušných dílků, ze kterých mohou skládat větší celky. Tato úroveň je velmi podobná úrovni předchozí, pouze s tím rozdílem, že výsledný obraz vytváří čaroděj díky svému kouzlení, a ne žák kurzorem myši. Ovládací panel na této úrovni je velmi omezen.



Obrázek 2 - Baltík - režim Čaruj scénu

Programuj – na této úrovni žák již opravdu programuje, jelikož se mu rozšíří možnosti ikonických příkazů a dalších parametrů nutných k programování. Žáci mají na výběr ze dvou režimů. V režimu *začátečník* je panel omezen pouze na základní příkazy, v *pokročilém* režimu se panel plně rozvine a zobrazí všechny možnosti, které program nabízí. Soubor příkazů Baltíka 3 zahrnuje všechny podmíněné příkazy (if, else if, switch- case), cykly (for, while, do-while), logické, relační a bitové operátory, proměnné, pole, procedury, rekurzi, příkazy pro práci se složkami a soubory, pro práci s datem a časem, pro práci s klávesnicí a myší, pokročilé grafické příkazy, matematické funkce atd. Pro ladění programů je k dispozici krokování, zobrazení proměnných. (SGP Baltík 3, 2017). Hlavně díky poslední funkci se z Baltíka stává plnohodnotný

programovací jazyk, jelikož u většiny „dětských“ programovacích jazyků je fáze ladění a krokování velmi omezena nebo úplně opomenuta.

Obrázek 3 - Baltík - ovládací panel: Programuj pokročilý

Že se jedná především o výukovou pomůcku programování, svědčí nepřeberné množství návodů dostupných na internetu a samozřejmě velmi dobře zpracovaná metodika výuky přímo na stránkách tvůrců.[15]

Pokud pro výuku programování využijete novou verzi Baltie 4 C#, bude Vaše prostředí obohaceno o třetí rozměr, což bude pro žáky přeci jen atraktivnější, bohužel výsledná grafika je velmi primitivní, a ve srovnání s 3D grafikou, kterou žáci znají z různých novodobých her, působí tato grafika velmi zastarale. Druhým bonusem je volba zobrazení C# kódu, kde mohou starší žáci porovnávat grafický zápis se skutečným objektově orientovaným zápisem v jazyku C#.

Přestože je tento programovací jazyk zatížen počítačovou licencí, nelze opomenout velmi vstřícný krok autorů směrem ke školství. V rámci výhodné multilicenční politiky lze využívat libovolnou verzi programovacího jazyku společnosti SGP Systems. Licence je bohužel časově omezena pouze na 1 rok.

[15]https://www.sgpsys.com/cz/

9.6 KODU Game Lab

Společnost Microsoft před nedávnem vyvinula jednoduché prostředí, ve kterém si lze během chvíle s minimálními znalostmi programovacího kódu vytvořit vizuálně působivé hry.

Tento program se lehce vymyká klasickému výukovému softwaru, jelikož jeho původní zaměření bylo cíleno pouze na tvorbu počítačových her. Z tohoto důvodu je ovládání programu navrženo pro herní ovladač stejnojmenné firmy. Přesto ho lze provozovat také v počítači.

Celý systém je postaven na výběru vhodného příkazu z nabídky dlaždic umístěných na kruhových výsečích, pomocí kterých žáci postupně skládají výsledný program. Na základě volby určitého nástroje se poté žákům otevírá nabídka s dostupnými možnostmi nastavení konkrétního nástroje, což velmi zjednodušuje tvorbu kódu. Celkem program obsahuje cca 120 dlaždic, které mohou žáci ke svému programování využít.

KODU obsahuje tři zásadní změny, které ho odlišují od programovacího jazyka Baltík.

První změnou je čistě objektový přístup, se kterým se žáci setkávají již při prvních zkušenostech s programem. Zvykají si na to, že každý objekt má své vlastnosti, kterými se od sebe stejné objekty (např. hvězdy) odlišují.

Druhou změnou je styl zápisu programového kódu, který se velmi nápadně podobá klasickému programování. Jednotlivé řádky kódu jsou uvozeny podmínkovým příkazem, který rozhoduje o tom, zda se daná část kódu bude vykonávat, či nikoliv. Tímto způsobem začnou žáci samovolně přemýšlet o jednotlivých akcích, které daný objekt vyhodnocuje a ne jen jako o posloupnosti příkazů. Zkrátka se zde žáci setkávají s řízeným průběhem programu.

Bohužel se žáci nenaučí, jak tento zápis v klasickém programovacím jazyce napsat. Na druhou stranu, vzhledem k věkovému zaměření, které cílí na mladší uživatele, není tato volba nezbytná.

Obrázek 4 - KODU Game Lab - detail programovacího prostředí

Třetím a rozhodujícím faktem ve školním prostředí je jeho dostupnost, což je velkou předností tohoto systému. Není vázán žádnou placenou licenční smlouvou. Součástí tohoto systému v ČR je navíc základní metodická příručka, která je vyvíjena v rámci stále trvajícího projektu Akademie programování[16], pomocí které mohou učitelé připravit celou hodinu (respektive několik vyučovacích hodin) díky čtyřem základním, dopodrobna zpracovaných programovacích tématům, k nimž mají pedagogové přístup po zaregistrování své školy do tohoto projektu. Díky této registraci získají zcela zdarma úvodní zaškolení zkušenými lektory. Tato aktivita vznikla na základě mezinárodního projektu Hour of Code (Hodina kódu)[17] , který nabízí didakticky zpracované vyučovací hodiny programování v několika zvolených programovacích jazycích, ve 45 cizojazyčných variantách včetně českého jazyka. Této metodiky mohou využít žáci také ke svému samostudiu a rozvíjení svých programátorských a algoritmizačních schopností.

KODU má také svá omezení vycházející ze zaměření systému, který je více orientován na grafickou stránku výsledné hry, méně na jeho funkční možnosti. Autor s tímto programem pracuje v rámci mimoškolní aktivity pro 3. až 5. ročník na Základní škole Ladova v Litoměřicích. V rámci této aktivity se při práci na projektech narazilo na funkční omezení, se kterými se musí v návrzích dalších programů počítat a je nutné je různými netradičními postupy alespoň částečně nahradit. Hlavním omezením, na které autor narazil, je tvorba proměnných, které v KODU nelze vytvářet. V programu si musíte vystačit s předem definovanými proměnnými, kterými jsou score a zdraví.

Výhodou je možnost velkého množství označení proměnných typu score, ze kterých si lze udělat pomocné proměnné, které lze využít k testování různých situací.

[16]http://www.akademieprogramovani.cz/
[17]https://hourofcode.com/cz/learn

Dle názoru autora je systém KODU ideální pro první krůčky na cestě k programování, jeho možnosti dalšího využití, hlavně u starších dětí, jsou značně omezeny.

10. Vybraní zástupci robotických stavebnic a programovatelných robotů

Programovatelní roboti a robotické stavebnice se vyznačují svou nezávislostí na počítači, díky němuž jí lze využívat i mimo klasickou informatickou učebnu. Tato práce však není zcela komfortní a pro složitější aplikace je téměř nepoužitelná, proto je tato oblast vyčleněna z metod, které jsou na počítači zcela nezávislé.

Vzhledem k rozvoji moderních technologií se do výuky algoritmizace a rozvoje programování čím dál tím častěji zapojují různé programovatelné elektronické hračky a stavebnice. Tato forma výuky je pro většinu žáků atraktivní, přesto se s ní mohou setkat pouze v rámci zájmových činností Domu dětí a mládeže s elektrotechnickým zaměřením. Je to z toho důvodu, že pořízení podobných pomůcek v dostatečném počtu je pro většinu základních škol velmi nákladné.

Jelikož existuje řada možností, jak se k těmto pomůckám dostat, budou popsáni hlavní představitele této kategorie výukových pomůcek.

10.1 LOGO

Přestože je tento trend na vzestupu nyní, kdy na trhu existuje několik variant robotických stavebnic a robotů, lze podobný přístup vypozorovat již v 70. letech.

Původcem těchto robotů byla mechanická želvička LOGO vyvinutá Tomem Callahanem na MIT v letech 1969-1970. Tento robot byl ovládán stejnojmenným programovacím jazykem. Celá koncepce se postupem času zdokonalovala, z původního kabelového přenosu informací se postupem času přešlo na bezdrátový přenos, a došlo k designové změně podoby robota. Součástí robota byla zápisová část, která mohla zaznamenávat trasu robota, čímž se z něj stal grafický nástroj. Ve spojení s touto koncepcí vešel do povědomí pojem *želví grafika.* Vzhledem k omezeným možnostem pohybu vykazuje tato grafika typické znaky. Stejné koncepce se využilo i v případě převodu do virtuálního prostředí.

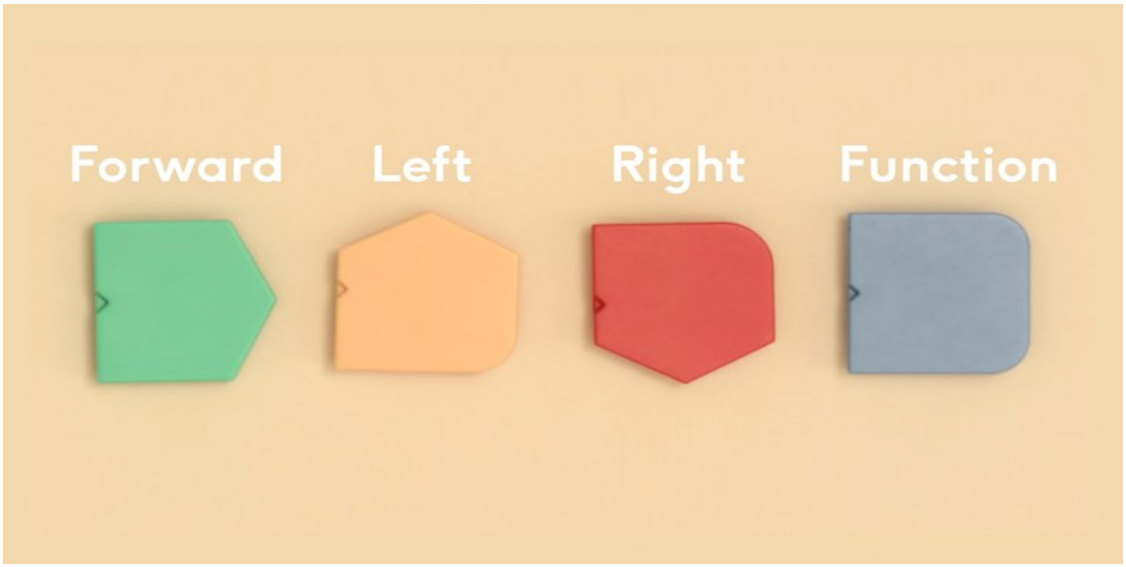
Přestože je tato koncepce velmi stará, navzdory tomu je této technologie využíváno dodnes v každém dálkově řízeném zařízení. Jediný rozdíl mezi původním a stávajícím pojetím je využití soudobé technologie, nabízející např. bezdrátový přenos.

Existuje celá řada podobných robotických „stavebnic“, většinu z nich lze využít také pro výuku programování u starších dětí, bohužel se ve všech případech jedná o placené řešení, na školní poměry většinou velmi finančně nákladné.

10.2 CUBETTO

„Cubetto je přátelský robot, který naučí vaše děti základům programování a logického myšlení bez použití počítače.“ (Malý programátor, 2017)

Zcela opačný přístup zvolili tvůrci systému CUBETTO, který k tvorbě programu ovládajícího dřevěného robota využívá speciální programovací panel, který je bezdrátově propojen se samotným robotem. Hlavní výhodou tohoto systému je samotný zápis programu, u kterého není podmínkou zvládnutí četby. Jednotlivé příkazy se provádí pomocí vkládání různobarevných dřevěných dílků do programovacího panelu. Příslušný příkaz je intuitivně reprezentován příslušným tvarem. Skládáním jednotlivých dílků žáci vytváří frontu příkazů, což je základním konceptem programování. Žáci mohou použít pouze 4 příkazy, z čehož jeden slouží pro pohyb kupředu, další dva ke změně směru a poslední využívá volání funkce, která se nachází ve spodní části panelu a obsahuje celkem čtyři pozice.



Obrázek 5 - CUBETTO- detail programovacích dílků

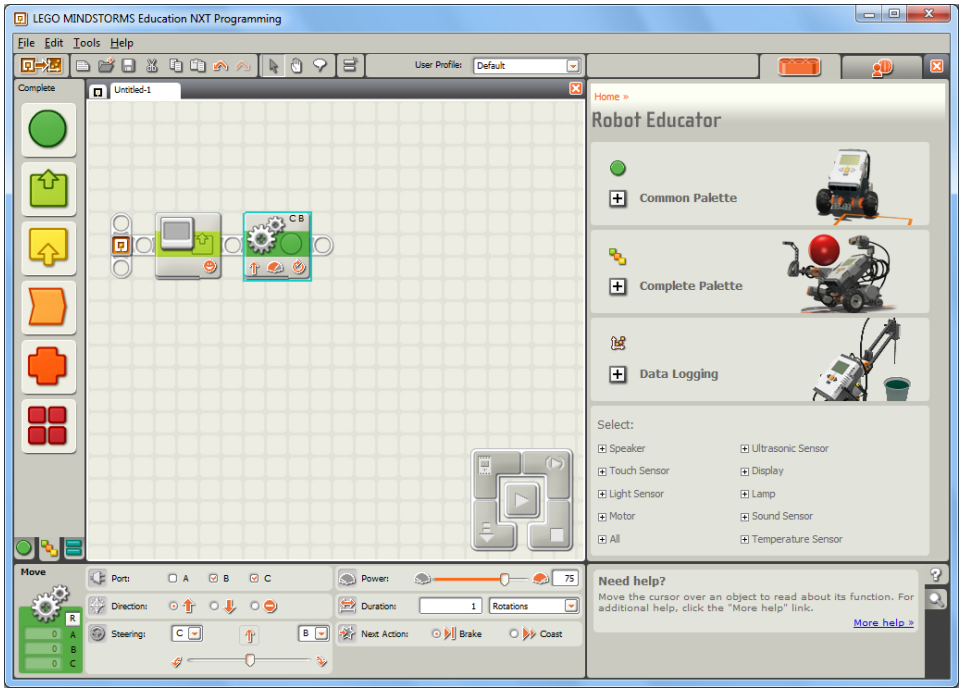
Robot ve tvaru dřevěné kostky se pohybuje po hracím plánu, jehož cílem je dostat se z jednoho místa hracího pole na druhé.

Celý systém je v dnešní době dostupný v několika českých obchodech.

10.3 LEGO Mindstorms

Jedním z podobných systémů, který je zároveň v tomto ohledu velmi rozšířený, je bezpochyby robotická stavebnice LEGO Mindstorms. Přestože lze k programování využít pouze vestavěné sady příkazů v centrální jednotce stavebnice, není tato manipulce příliš komfortní, a mnohem snadněji lze stavebnici programovat skrze počítačový program nabízející mnohem širší škálu dostupných příkazů.

Základem celé stavebnice je řídící jednotka tzv. *kostka*, která vyhodnocuje informace z připojených senzorů, díky nímž řídí aktivní prvky stavebnice. Pro jednodušší programování lze využít volně dostupné programové prostředí, které je postaveno na stavebnicové koncepci a je silně intuitivní.



Obrázek 6 - Programovací prostředí

Každý blok příkazů má dodatečná nastavení, která jsou umístěna v dolní části prostředí, sloužící k přesnějšímu nastavení jednotlivých senzorů. Bohužel citlivost některých senzorů je největším úskalím, na které se musí při stavbě robota pamatovat. Patří sem různé druhy fotosenzorů, jejichž správné vyhodnocování velmi významně ovlivňuje osvětlení okolního prostředí.

Předností tohoto systému je možnost volby programovacího jazyka. Pro různé programovací jazyky lze nahrát do kostky pomocí speciálního firmwaru příslušný překladač. Další výhodou je možnost univerzálního propojení různých LEGO Mindstorms a jeho prvků, stejně jako je zvykem u klasické LEGO stavebnice.

Hlavní překážkou masivního rozšíření na běžném typu základních škol a jejich využití pro výuku algoritmizace a programování je bezesporu její pořizovací cena, která přesahuje hodnotu 10 000 Kč. Přes to všechno je tato stavebnice hojně využívána v základním školství se zaměřením na IT technologie. Jedním z hlavních využití tohoto systému je práce v různých zájmových činnostech v mimoškolních aktivitách, pomocí nichž lze stavět nejen funkčně působivé, ale také vizuálně atraktivní samočinné jednotky.

10.4 Lego WeDo v2.0

Nejnovějším přírůstkem do rodiny robotických hraček je Lego WeDo v2.0, kterému se krátce věnuje tato kapitola.

Hlavním důvodem vzniku systému Lego WeDo v2.0 je jeho podobnost se systémem Lego Mindstorms. Pro počáteční seznámení s touto robotickou hračkou není nutné hned pořizovat komplexní systém, jakým Lego Mindstorms bezesporu je, ale naprosto postačí právě systém Lego WeDo v2.0

Hlavní rozdílem jsou menší možnosti sestavení robota, jelikož je řídící jednotka[18] osazena pouze dvěma vstupy. Také samotné senzory nemají tak bohatou nabídku jako v případě Lego Mindstorms. WeDo ve své sadě nabízí pouze dva senzory, a to pohybový senzor a senzor náklonu.

[18] Řídící jednotka systému Lego WeDo je označována jako Smart Hub. Doplňkovou komponentou je dobíjecí baterie, která se s řídící jednotkou spojí. Díky ní je zajištěno napájení na několik hodin.

Obrázek 7 - SmartHub s napájecí jednotkou

Systém Lego WeDo je primárně zaměřen na mladší školní věk a jeho hlavním cílem je prvotní seznámení se světem robotiky a základy programování robotů. Myšlenka je podpořena designovými doplňky stavebnice, které dělají z robotů více „dětsky“ vypadající roboty. Dalo by se říci, že se jedná o zjednodušenou verzi předchozího systému. Ovšem i přes všechna tato omezení lze postavit mnoho rozličných robotů.

Obrázek 8 - Vývojové prostředí WeDo 2.0

Rozsáhlou galerii nápadů a návodů ocení díky metodicky zpracovaným pracovním listům nejen pedagogové, ale i samotní žáci jako pomůcku při svém samostudiu.

K programování řídící jednotky lze použít programové prostředí, které je volně dostupné na stránkách výrobce.

Toto prostředí je jednoduché a pro děti intuitivní díky názornému zobrazení jednotlivých ovládacích prvků. Jedinou nevýhodou při použití v českých školách je absence české lokalizace vývojového prostředí. Tento nedostatek však dostatečně kompenzuje grafické zpracování, které je názorné a nápomocné při sestavování programu. Užitečná je naopak integrace knihovny možných projektů s podrobným návodem na stavbu a nastavení robota.

Další možností je využití velmi oblíbeného programového prostředí SCRATCH, do kterého lze doinstalovat doplněk[19], který přidá speciální sadu příkazů k ovládání komponent WeDo.

Propojení vývojového prostředí a řídící jednotky robota je zajištěno díky bluetooth technologii, díky které je možno chování robota ovlivňovat v reálném čase a při vývoji aplikace není nutné robota složitě připojovat pomocí USB kabelu. Tato možnost nabízí variantu použití jedné stavebnice ve třídě, kdy jednotlivé vývojové týmy vytvářejí svou aplikaci na PC a poté ji při otestování nahrají do řídící jednotky. Koncept lze použít pouze za předpokladu, že všichni žáci vyvíjejí aplikaci pro předem postaveného robota, s jasně daným omezením funkčnosti a možnostmi použití. Tento přístup vytváří kreativní prostředí pro práci vývojových týmů. Má však i jeden nedostatek, kterým je možnost vyzkoušení funkčnosti aplikace v průběhu vývoje, vzhledem k počtu participujících skupin.

10.5 Ozobot

Tento malý robot, kterého lze již dnes bez problémů pořídit v několika českých obchodech, je ve skutečnosti „sledovač“ čáry. Sledování čáry je ve své podstatě jednoduchá úloha. Obtížnou se stává v případě, že je nutný plynulý a rychlý pohyb robota po vytyčené trase.

Robot je řízen pomocí čar čtyř barev, konkrétně černé, modré, zelené a červené, kdy každá barva konkrétním způsobem mění chování robota.

[19] https://scratch.mit.edu/wedo

Barvy jsou z herního plánu načítány díky soustavě pěti světelných čidel, které jsou umístěny vpředu ve spodní části podvozku.

Obrázek 9 - Detail podvozku ozobota s fotosenzory

Pokud nakreslíme soustavu několika konkrétních barev v úsecích dlouhých 5 – 7 mm pro každou z nich, získáme příslušný příkaz, díky kterému můžeme velmi jednoduše ovlivňovat pohyb ozobota po námi nakreslené trase. Tato trasa nemusí být nutně nakreslená pouze na papíře, ale v době moderních technologií lze ke kresbě trasy využít různá zobrazovací dotyková zařízení jako je např. tablet.

Kromě využití tzv. *ozokódu*, který vzniká příslušnou sekvencí barevných značek, lze robota také programovat pomocí multiplatformního aplikačního prostředí, ve kterém jsou využity předdefinované nabídky příkazů rozdělených do logických celků, díky nimž je možné programovat pohyb robota bez přítomnosti vodících čar na ploše.

Obrázek 10 - Ukázka ozokódu tvořený barevnou sekvencí

Pokud si vyberete ovládání robota formou vyznačené trasy na papíru, je třeba zvolit vhodné fixy splňující potřebné odstíny barev, které dokáží čidla správně interpretovat. Přestože k novější verzi ozobota, kterým se stal ozobot evo, jsou dodávány speciální popisovací fixy, lze využít i fixy běžně dostupné v obchodě. Nejvhodnějším typem popisovačů jsou fixy od společnosti Centropen. Pro záznam černé a červené barvy se ukazují nejvhodnější permanentní popisovače společnosti Centropen. Záznam zelené a modré barvy je lepší pořídit pomocí popisovačů na stíratelnou tabuli, neboť u permanentních fixů těchto barev jsou jejich odstíny příliš tmavé

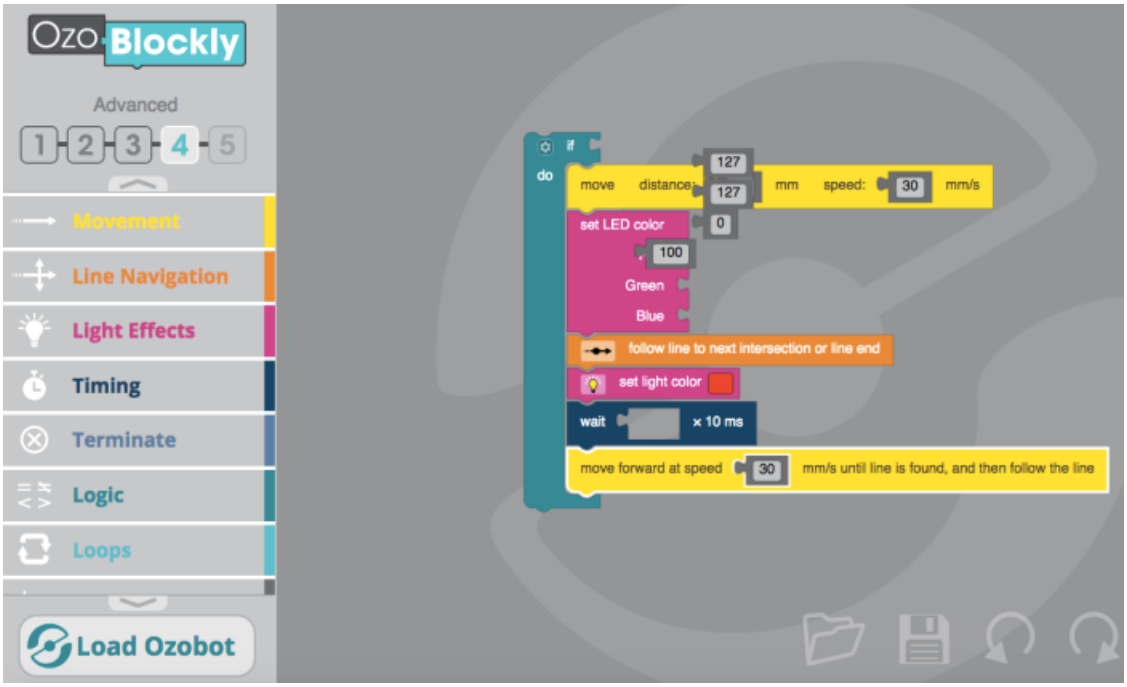
Kromě správného odstínu je pro správné vyhodnocení trasy nezbytné dodržet minimální šířku čáry o rozměrech 5 – 6 mm. Z tohoto důvodu je výhodné použít fixy se skoseným hrotem vytvářejícím silnější stopu. U popisovače se špičatým hrotem je možné k zvýraznění použít dvojitou čáru.

Pro kreslení čar lze využít aplikačního prostředí OzoDraw, které podporuje všechny platformy mobilních OS. Obrazovky tabletu lze využít k pohybu ozobota.

Toto prostředí nabízí kompletní přehled všech příkazů, které lze libovolně umisťovat na vytvořenou trasu.

Druhou variantou programování ozobota je využití programovacího prostředí *OzoBlockly*, který do ovládání robota vkládá konstrukce známé z klasického programování. Prostředí je nápadně podobné prostředí programovacího jazyku Scratch, což usnadňuje zařazení tohoto prvku do výuky programování v případě znalosti Scratche.

Samotné převedení hotového programu do robota probíhá bezdrátově, a to přiložením spodní části robota na vyznačené místo na obrazovce. Takto naprogramovaný robot může plnit mnohem sofistikovanější úkony, kterých nelze dosáhnout pouhým zápisem grafických značek.



Obrázek 11 - Programové prostředí OzoBlockly k programování robota

Programové prostředí je rozděleno na logické celky, jak je zobrazeno na obrázku č. 11, díky nimž se dá ovládat nejen pohyb robota, ale také ovlivňovat světelné efekty tvořené soustavou RGB LED diod umístěných v horní části robota pod průhledným krytem.

Ozobota lze pořídit v několika verzích lišících se od sebe převážně senzorovým vybavením, využívaných při jejich programování. Základní verze Ozobot v2.0 obsahuje

pouze pětici světlocitlivých čidel umístěných na podvozku robota, nová verze Ozobot Evo je rozšířena o sedm čidel a senzorů, které nově detekují situaci nejen pod ním, ale také kolem robota. Toto chování je zajištěno díky vnitřní inteligenci robota. Evo například dokáže rozpoznat překážku před ním a zastavit, aniž by do ní naboural.

Jelikož tvůrci ozobota cílí na vzdělávací instituce, ne na domácnosti, vytvořili pro pedagogy rozsáhlý portál [20] shromažďující rozličné aktivity a nápady pro výuku. K inspiraci a jako zdroj dalších informací o této didaktické pomůcce lze využít také české stránky, zabývající se využitím Ozobota v edukačním procesu.[21]

Předností této koncepce je možnost libovolného vizuálního rozšíření, což lze realizovat formou aktivity v hodinách pracovních činností, a vytvořit tak zajímavou mezipředmětovou vazbu. Na robota lze „navléknout" libovolný vzhled vytvořený z papíru, který nemá vliv na funkčnost, neboť jediná funkční část ve spodní části robota není převlekem dotčena. Konstrukce je natolik lehká, že jeho hmotnost neovlivňuje pohyb robota. Na internetových stránkách výrobce existuje bezpočet předloh, pomocí nichž lze vzhled ozobota změnit k dokreslení výsledného efektu[22].

10.6 Bee-bot

Na konec této kapitoly se vrátíme k původní myšlence mechanické želvy LOGO a představíme si nápadně podobnou edukační interaktivní pomůcku nazvanou Bee-bot, neboli robotická včelka.

Jedná se o mechanického robota, který je přímo ovládaný a programovatelný. K programování se využívá tlačítek umístěných na zádech včelky, pomocí nichž lze předem určit její pohyb.

[20]http://portal.ozobot.com/lessons
[21] http:// http://ozobot.sandofky.cz/
[22]http://ozobot.com/play/print-games

Obrázek 12 - Detail na ovládací prvky

Základem této didaktické pomůcky je čtvercová síť o rozměru jednoho čtverce 15x15cm. Právě tato vzdálenost je tzv. krok včelky. Na základě karet různého zaměření nebo tematických podložek lze jejího pohybu využít v různých hodinách, tedy i v předmětech neinformatického zaměření.

Hlavním omezením zapojení této pomůcky do složitějších aplikaci je její vnitřní paměť, která dokáže pojmout pouze 40 kroků, což úzce souvisí s velikostí podložky.

11. Herní prostředí

Při výuce má herní přístup z pohledu dětí největší přínos, jelikož k edukaci dochází v rámci jejich přirozeného a dobře známého prostředí, a proto k němu přistupují s nadšením. Mnohdy ani nemusí postřehnout, že k edukaci dochází, vymyslet řešení určitého problému je bráno jako součást celé hry. Tato prostředí však mají svá velká úskalí. Tím prvním je finanční omezení školy, protože převážná většina podobně orientovaných programů jsou licenčně vázány. Druhým úskalím je časové omezení výuky, jelikož v rámci nízkých hodinových dotací je třeba velkou část informací žákům předat a nelze vždy „čekat“, až tyto vztahy a zákonitosti v rámci hry objeví. Přesto je to velmi vhodné zpestření hodin, které žáci vřele uvítají. Nejznámějším a hlavně v našich školách nejvíce rozšířený je Minecraft, proto mu bude věnována následující podkapitola.

11.1 Minecraft

Minecraft se do škol dostal v roce 2016 díky speciální edici Minecraft:Eduction Edition. Tuto edici je možné v rámci školního prostředí zdarma vyzkoušet, nicméně integrace Minecraftu do výuky je již zpoplatněna. Do vývoje této edice se v rámci testování zapojilo přes 50 000 studentů a učitelů.

Základem této hry je tvorba rozličných objektů z předem natěžených surovin. Tato strategie je však nemyslitelná ve školním prostředí díky zdlouhavé fázi edukace, a proto se v edici Education od tohoto principu ustoupilo a žáci mají k dispozici neomezené zdroje. Mohou tedy bez jakýchkoliv zdržení začít řešit zadané úlohy, které přímo v tomto prostředí vytváří učitel.

Úkolem žáků je na základě posloupnosti instrukcí zautomatizovat práci, kterou v běžné herní edici řeší většinou manuálně.

12. Pedagogická opora výuky programování

V této kapitole je představen metodický portál pro pedagogy, který shromažďuje všechny výše popsané přístupy na jednom místě.

Tímto portálem je Kidscodr[23], který nabízí rozličné výukové kurzy formou videotutoriálů, které jsou pro žáky nejnázornější. Do těchto kurzů se může žák zaregistrovat zdarma, např. pomocí google účtu nebo účtu na Facebooku.

Veškeré kurzy jsou rozděleny do dvou částí. V první části je žákovi ukázána příslušná programová dovednost, kterou později využije při plnění úkolu, který je žákovi zadán vždy na konci každého bloku kurzu. V rámci registrace do kurzu získá žák veškeré potřebné materiály, které pro splnění úkolu potřebuje. Díky tomu se žáci mohou věnovat pouze programové části zadaného úkolu, což velmi pozitivně ovlivňuje časovou náročnost splnění úkolu.

Velkým přínosem je možnost volby rozličných vývojových prostředí, které lze volit na základě náročnosti, atraktivnosti pro žáky a technického vybavení školy. Jedním z atraktivnějších výukových kurzů je tvorba aplikací pro mobilní platformy.

[23] https://www.kidscodr.cz/

13. Zpracování dotazníkového šetření v rámci ZŠ

Dotauník zmiňovaný v této kapitole je dostupný na https://drive.google.com/open?id=1s5tTbDBFdSvC5-ML_xbVZkk0BwXtzikA_EfZxZx9NhQ.

Vzhledem k zaměření práce na zapojení výukových metod a programových prostředí rozvíjejících informatické myšlení dotázaní pedagogové odpovídali na otázky, jakou měrou těchto možností využívají. Jelikož již v letech 2006 a 2013 došlo k rozsáhlému výzkumu, bude součástí analýzy provedeného dotazníkového šetření jejich přímé porovnání.

Dle výzkumu prováděného v letech 2006 až 2007, na který je odkazováno v první části práce, se ukazuje, že informaticky zaměřený předmět není ve srovnání s hlavními vyučovacími předměty považován za rovnocenný, a tudíž se na něj neklade dostatečný důraz.

Tento stav je dán především nízkým počtem pedagogů s aprobací na informatiku. Jelikož informatiku učí většinou pedagogové s nejbližším zaměřením, podřizují náplň výuky svým znalostem a dovednostem. Tento jev vede k očekávanému závěru, tedy že výuka je zaměřena hlavně na uživatelské dovednosti, a do pozadí se dostává tvůrčí činnost, která by u žáků rozvinula algoritmické myšlení.

13.1 Distribuce a popis respondentů

K distribuci dotazníku bylo použito dvou elektronických komunikačních kanálů. Jedním z kanálů byla emailová korespondence adresovaná vedení základních škol. Druhým komunikačním kanálem se stala sociální síť Facebook, konkrétně facebooková skupina „Pedagogická komora“ a „Pedagogické.info“.

Přestože emailová korespondence nesplnila očekávání, Facebook se stal správnou volbou. Tato varianta sebou, jak se později ukázalo při analýze odpovědí, přinesla jistá úskalí.

Respondenty se stali aktivní členové těchto skupin, bez ohledu na jejich zaměření. Jelikož informatické myšlení není spjato pouze s výukou informatiky, mají odpovědi zcela jedinečnou vypovídající hodnotu.

13.2 Struktura dotazníku

Dotazník je rozčleněn do dvou částí, které jsou vybrány na základě klíčových dotazů. Ze začátku dotazníku je respondentům položeno několik společných otázek, ilustrující jejich zaměření, stupeň ZŠ, na kterém nejvíce vyučují a kraj, ve kterém se jejich škola nachází.

Po rozdělující otázce, zda respondent vyučuje informatiku či nikoliv je dotazník rozvětven na dvě nezávislé části.

V části pro informatiky je zkoumána délka praxe a hodnocení stávajícího pojetí vzdělávací oblasti Informační a komunikační technologie v rámci RVP ZV a hodnocení důležitosti tematických celků.

Poslední dvě kategorie byly zvoleny převážně díky možnému srovnání s výzkumy VIV06 a VIV13.

Další oblasti zjišťují dobu výuky informatiky v rámci základního vzdělání a používané programové prostředí a výukové metody, vedoucí k rozvoji informatického myšlení.

Část pro ostatní učitele zkoumá informatické myšlení v neinformatických předmětech a jeho využití při výuce.

13.3 Metody zpracování dat

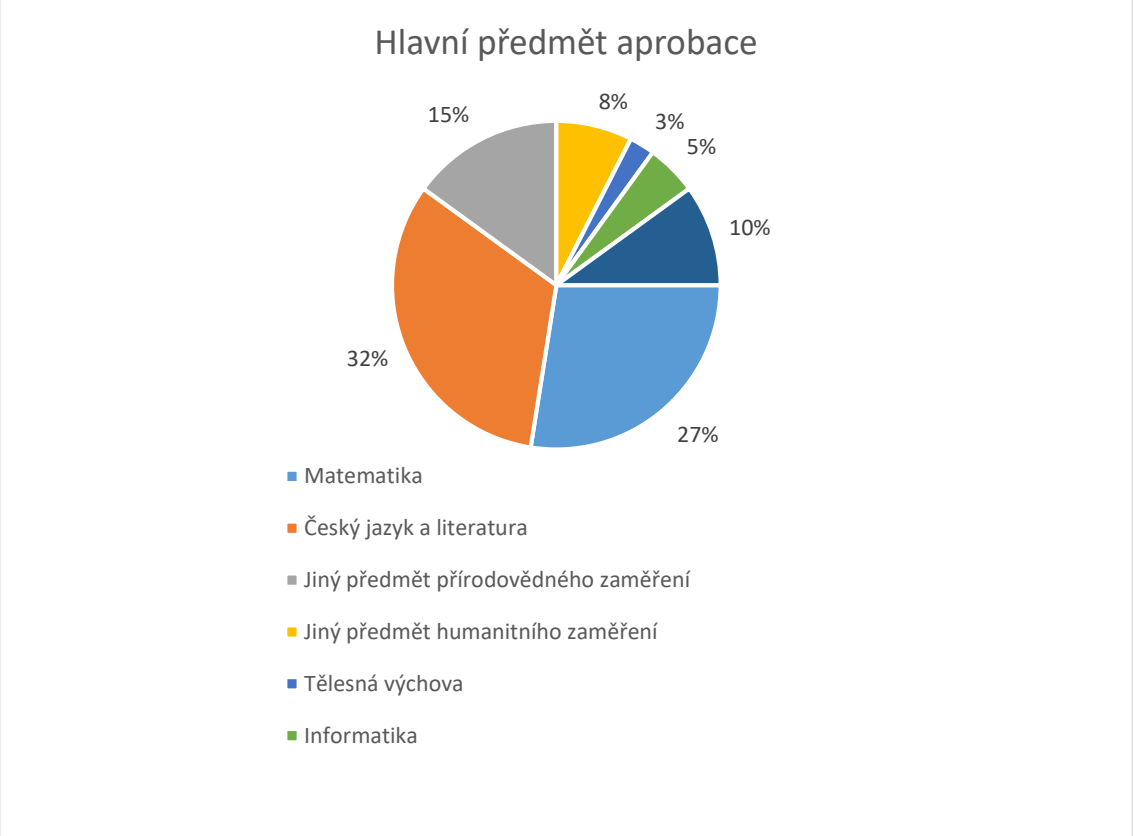
Vzhledem k menšímu počtu respondentů, než bylo původně očekáváno, a vzhledem k charakteru odpovědí, byly odpovědi zpracovány v rámci četnosti stejných odpovědí. Z těchto údajů byly poté vytvořeny grafy, ilustrující stávající situaci dané oblasti výzkumu.

V případě otevřených otázek byla zvolena metoda shluků odpovědí podobného významu. Tento typ otázek však neměl příznak povinné odpovědi, proto mnoho respondentů takovéto otázky nevyplňovalo.

13.4 Závěry dotazníkového šetření

Veškeré grafy, které jsou zobrazeny v této kapitole, vznikly na základě zpracování dotazníkového šetření.

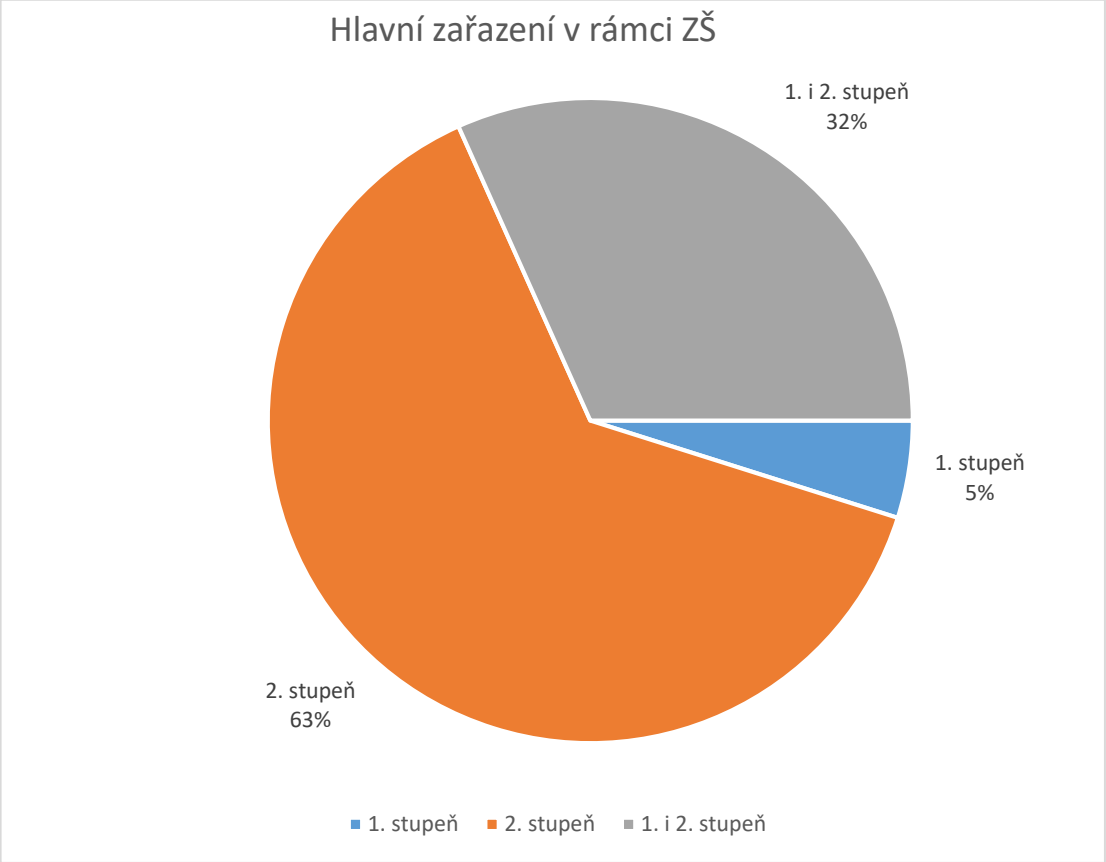
Jak již bylo zmíněno výše, rozložení respondentů bylo lehce neočekávané. Kromě očekávaného podílu matematiků 27,5% se objevila velká skupina učitelů s aprobací Český jazyk a literatura, která tvořila 32,5% všech respondentů. Tento výsledek lze nejspíše vysvětlit zvýšenou aktivitou takto zaměřených pedagogů ve zmiňovaných



Graf 1 - Rozdělení respondentů dle hlavního předmětu aprobace

facebookových skupinách. Druhým pozoruhodným, ne však neočekávaným, závěrem je počet pedagogů s nedokončeným vysokoškolským vzděláním pedagogického směru čítající v analyzovaném vzorku respondentů 10%. Tato hodnota je mírně zvýšena oproti posledním výzkumům, není však zcela nereálná.

Z výzkumu dále vyplývá, že většina dotázaných pedagogů považuje rozvoj informatického myšlení za velmi prospěšný a důležitý, bohužel však dodávají, že na takovýto způsob výuky mají nízkou hodinovou dotaci daného předmětu.



Graf 2 - Rozdělení respondentů dle působnosti na ZŠ

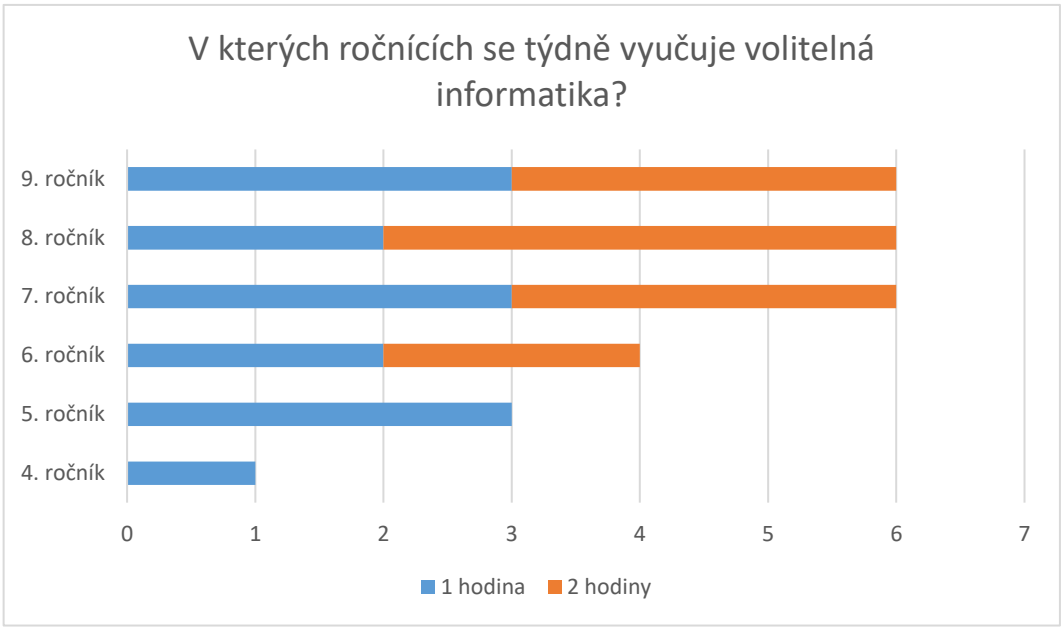
Co se týče rozložení výuky informatiky v rámci základního vzdělání, ukázalo se, že téměř polovina škol z daného vzorku respondentů využívá pouze dvouhodinovou dotaci na celý vzdělávací cyklus, tedy jednu hodinu v rámci každého stupně základní školy. Ve zbylých školách je hodinová dotace na výuku informatiky navýšena o disponibilní hodiny. Ve čtvrtině škol probíhá informatika v každém ročníku druhého stupně ZŠ, ve zbylé čtvrtině je tato možnost využita pouze v některých ročnících druhého stupně.

Z pohledu zastoupení jednotlivých celků jednoznačně vyplývá, že ve výuce je stále kladen důraz na uživatelské dovednosti, což prokázal výzkum VIV06. Pochopitelný je zájem o bezpečnou komunikaci na sociálních sitítch a zabezpečení počítače jako takové. V neposlední řadě se také rozšiřuje zájem o algoritmizaci a programování. Bezmála polovina respondentů ji považuje za důležitou, druhá polovina by tuto oblast přesunula až na SŠ či nemají dostatek času na integraci tohoto celku do tematických plánů.

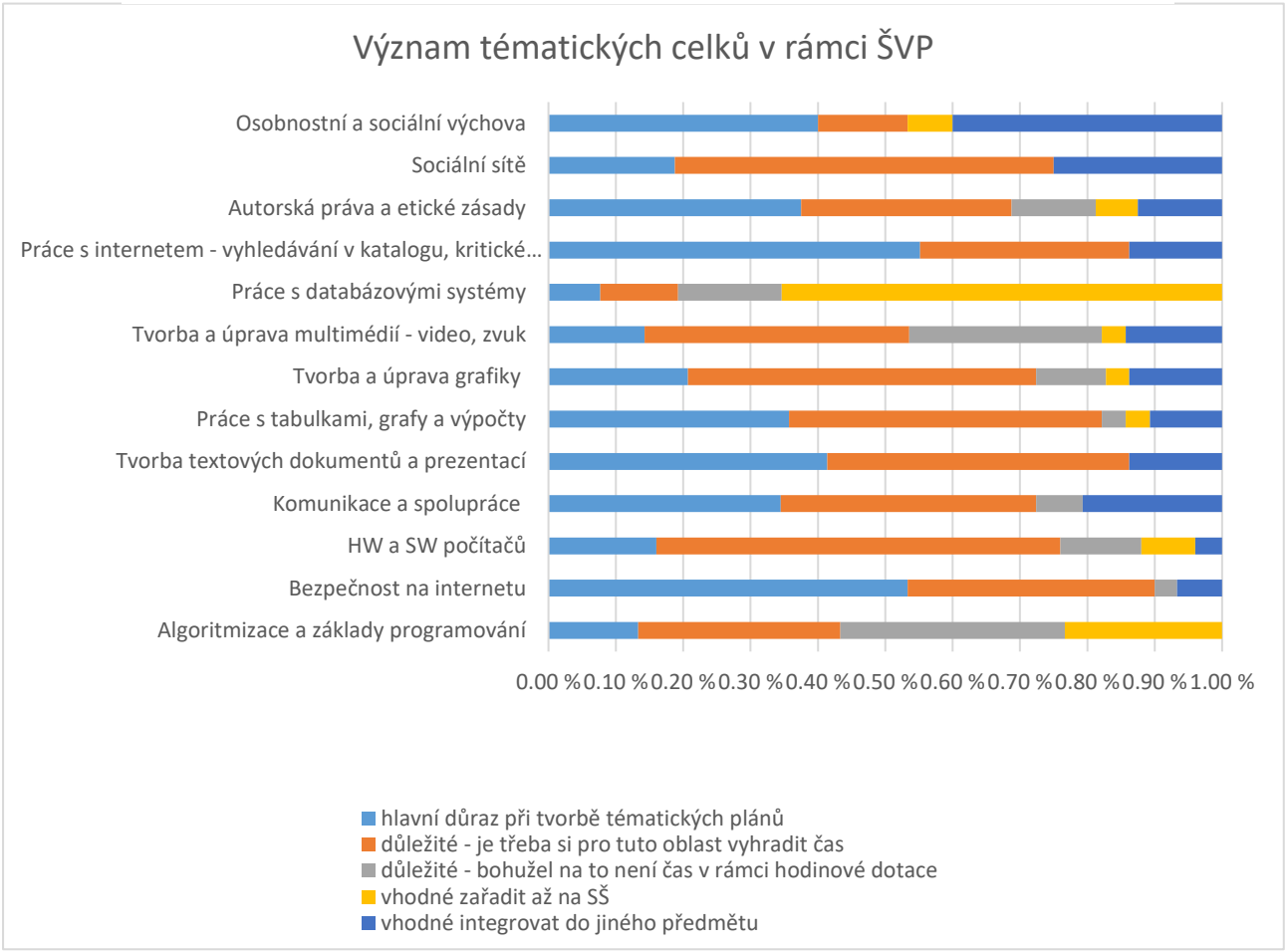
Graf 3 - Využití disponibilních hodin v rámci výuky informatiky

V porovnání s výzkumem VIV06 je to výrazný posun k lepšímu, jelikož v předchozích letech považovala většina respondentů tuto oblast za okrajovou nebo vhodnou až pro středoškolské vzdělání.

Vzhledem k nižšímu počtu respondentů jsou však tyto výsledky pouze orientační, a pro konkrétnější závěry by bylo třeba mnohem více respondentů.



Graf 4 - Výuka informatiky v rámci volitelného předmětu



Graf 5 - Význam tématických celků v rámci ŠVP

14. Závěr

V této práci, na základě v minulosti pobíhajících výzkumů, jsem zmapoval pozici výuky algoritmizace a programování v prostředí základních škol, která by se měla stát nedílnou součástí výuky informatiky vedoucí k rozvoji informatického myšlení.

Přestože jsem v práci vycházel ze závěru, že tato oblast je do dnešní doby spíše okrajovou záležitostí, potvrdil jsem si tuto hypotézu v rámci vlastního dotazníkového šetření, jehož závěry jsou v práci zmíněny.

Na základě těchto výzkumů jsem provedl analýzu různých metod, na podporu informatického myšlení u žáků, aniž bychom je hned ze začátku výuky od tohoto procesu odradili nevhodnými metodami nebo neatraktivním způsobem výuky.

V rámci této teoretické části jsem dohledal k popisovaným metodám vhodné zástupce, kde jsem z jisté části čerpal na základě vlastních zkušeností s těmito prostředími v rámci dlouhodobého vedení zájmové činnosti se zaměřením na algoritmizaci a programování.

V neposlední řadě jsem v praktické části vypracoval pracovní listy a metodiky k aktivitám, ve kterých se zabývám tzv. unplugged metodami, které mám díky vedení zájmové činnosti osobně ověřeny.

15. Metodika k pracovním listům

Tato kapitola obsahuje zpracované metodiky ke dvěma aktivitám, které vycházejí z ověřených postupů aplikovaných v rámci vedení zájmového kroužku se zaměřením na programování.

15.1 Grafické programování na papíře

Třída:

Časová dotace aktivity:

Cíle výuky:

- Žák sestaví kód pomocí předem dohodnutých symbolů
- Žák vytvoří vlastní kód pro spolužáka
- Žák chápe obtížnosti převodu reálných problémů do programu
- Žák si uvědomí rozdílný přístup v myšlením člověka a v interpretaci stejného algoritmického postupu počítačem

Prekoncept:

- žák využije informací z výkladu pedagoga a vlastní znalost problému např. z počítačových her

Pomůcky:

- pracovní list s připravenou mřížkou 4x4
- předloha s vytvořenými úkoly

Postup:

- Na začátku této aktivity učitel vysvětlí žákům pojmy algoritmus a program.
- Učitel žákům rozdá pracovní listy s „hracími poli“ a předem dohodnuté používané symboly programového kódu (viz příloha č. 1a a č. 1c).
    - Při volbě rozsahu symbolů je třeba mít na paměti, že větší počet symbolů dává žákům větší variabilitu při řešení daného problému, díky níž je hledání správného řešení jednodušší.
    - Naopak zúžením tohoto výběru jsou žáci nuceni ke stavbě optimalizovaného programového kódu.
- Žák převede příslušný obrazec do symbolického zápisu.

    - Pokud žák objeví v symbolickém zápisu opakující se části, přepíše tento zápis v kombinaci číslice a příslušného symbolu (pro žáka je důležité si tento postup osvojit a zapamatovat, bude využit jako prekoncept v úvodu výuky o cyklech).
- Žák předá tento zápis jinému žákovi, který na základě sepsaného postupu, který dodržuje, sestavuje stejný obrazec.

Poznámka:

- Výhodou této aktivity je možnost jejího využití i mimo hodiny informatiky, protože nevyžaduje použití počítače.
- Ná základě zadané předlohy lze vytvořit obdobné nebo i větší předlohy.
- Obdobnou metodu můžeme použít i v případě jiných povolených příkazů (např. je možné povolit pouze dva příkazy – o krok dopředu a otočení doprava).

15.2 Lidský automat

Třída:

Časová dotace aktivity:

Cíle výuky:

- Žák se naučí používat předem připravené příkazy
- Žák využívá informačního myšlení při tvorbě sekvencí příkazů

Prekoncept:

- Žák využije své znalosti z počítačových her

Pomůcky:

- min. 6 plastových kelímků
- dřevěné hůlky nebo špejle

Postup:

- Kelímky jsou vyskládány do řady na lavici nebo stůl.
- Žáci vytvoří dvojice a rozdělí si role – jeden se stává operátorem a druhý robotem.
- Operátor s použitím předem dohodnutých příkazů ovládá „robota“. Operátor tvoří sekvenci příkazů, na základě které robot mění svou pozici. Cílem je, aby robot postavil pyramidu z kelímků.
- Žák v roli robota, na základě operátorových příkazů, přemisťuje pomocí hůlek jednotlivé kelímky.

16. Seznam použitých zdrojů

[1] 1969 – The Logo Turtle – Seymour Papert et al (Sth African/American). *Cybernetic zoo.com: a history of cybernetic animals and early robots* [online]. [cit. 2017-07-07]. Dostupné z: http://cyberneticzoo.com/cyberneticanimals/1969-the-logo-turtle-seymour-papert-marvin-minsky-et-al-american/

[2] 2. Algoritmizace - Stránky k výuce informatiky [online]. [cit. 2018-07-13]. Dostupné z: http://www.ivt.mzf.cz/algoritmizace-a-programovani/uvod-do-algoritmu/2-algoritmizace/

[3] [online]. In: . [cit. 2017-07-07]. Dostupné z: http://old.spsemoh.cz/vyuka/algor/algritm.htm

[4] BELL, Tim, Ian H WITTEN, Mike FELLOWS, Robyn ADAMS a Jane MCKENZIE. Computer Science Unplugged: An enrichment and extension programme for primary-aged children [online]. 2006. Dostupné z: http://csunplugged.org/

[5] BROMOVÁ, Jana. Výuka algoritmizace na základní škole - aktuální stav. Č. Bud., 2012. diplomová práce (Mgr.). JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH. Pedagogická fakulta

[6] ČERNOCHOVÁ, Miroslava. *Rozvoj informačně technologických kompetencí na základních školách: výzkum stavu a struktury informačně technologické gramotnosti*. v Praze: České vysoké učení technické, 2013. ISBN 978-80-01-05407-9.

[7] DOSTÁL, Jiří. *Badatelsky orientovaná výuka: kompetence učitelů k její realizaci v technických a přírodovědných předmětech na základních školách*. Olomouc: Univerzita Palackého v Olomouci, 2015. ISBN 978-80-244-4515-1.

[8] FILIP, Brož. *Ozobot Evo je mnohem lepší než předchůdci, umí také komunikovat* [online]. In: . [cit. 2017-07-07]. Dostupné z: http://jablickar.cz/ozobot-evo-je-mnohem-lepsi-nez-predchudci-umi-take-komunikovat/

[9] HAVELKOVÁ, Hana. [online]. In: . [cit. 2017-07-07]. Dostupné z: http://wvc.pf.jcu.cz/ki/data/files/93prg1.ppt

[10] Hlavní zjištění výzkumu. [ZPRACOVAL KOLEKTIV ŘEŠITELŮ POD VEDENÍM VLADIMÍRA RAMBOUSKA]. Výzkum informační výchovy na základních školách. Plzeň: Koniáš, 2007, s. 14. ISBN 8086948102.

[11] HOFERKOVÁ, Kateřina. [online]. In: . [cit. 2017-07-07]. Dostupné z: https://www.svethardware.cz/hra-cubetto-nauci-programovat-i-ty-nejmensi/42049

[12] KOUBKOVÁ, A; PAVELKA, J. *Úvod do teoretické informatiky*. 3. vyd. Praha: MATFYZPRESS, 2003. ISBN 80-86732-03-7

[13] KVASIL, Bohumil, et al. Algoritmus. In Malá československá encyklopedie. Praha: Academia, 1984. s. 108.

[14] LESSNER, Daniel. *HOW DO WE TRANSLATE COMPUTATIONAL THINKING INTO CZECH?* [online]. In: . str. 1 [cit. 2018-04-20]. Dostupné z: http://ksvi.mff.cuni.cz/~lessner/w/data/_uploaded/file/papers/2014_02_lessner_didactig.pdf

[15] LESSNER, Daniel a Jiří VANÍČEK. Bobřík učí informatiku. Matematika – fyzika – informatika [online]. Praha: Prometheus, 2013, vol. 22, no. 5, pp. 374–382. Dostupné z: http://mfi.upol.cz/index.php/mfi/article/view/92/ 105

[16] *Malý programátor* [online]. [cit. 2017-07-07]. Dostupné z: https://www.malyprogramator.cz/

[17] MOTYČKA, Arnošt. Algoritmizace. Brno: Konvoj, 1999. ISBN 80-85615-80-0.

[18] MÜLLER, Karel. *Programovací jazyky.* První vydání. Praha: ČVUT, 2002. 219 s.

[19] Operational Definition of Computational Thinking for K-12 Education [online]. B.m.: International Society for Technology in Education (ISTE) a Computer Science Teachers Association (CSTA). 2011. Dostupné z: http://csta.acm.org/Curriculum/sub/CurrFiles/Co mpThinkingFlyer.pdf

[20] PITNER, Tomáš. *Výuka programování na základní a střední škole* [online]. In: . [cit. 2017-07-07]. Dostupné z: http://www.fi.muni.cz/~tomp/semuc/text_pitner.html

[21] *Primo* [online]. [cit. 2017-07-07]. Dostupné z: https://www.primotoys.com/

[22] Propedeutika. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 25. 8. 2016 [cit. 2017-07-06]. Dostupné z: https://cs.wikipedia.org/wiki/Propedeutika

[23] PRŮCHA, Jan. WALTEROVÁ, Eliška a Jiří MAREŠ. *Pedagogický slovník*. 7., aktualiz. a rozš. vyd. Praha: Portál, 2013, s. 185. ISBN 978-80-262-0403-9.

[24] PŠENČÍKOVÁ, Jana. Algoritmizace. Kralice na Hané: Computer Media, c2007.

[25] SAMKOVÁ, Libuše, 2011. Badatelsky orientované vyučování matematiky. In: Sborník 5. konference Užití počítačů ve výuce matematiky. České Budějovice: JČU, s. 336–341. ISBN 978-80-7394-324-0.

[26] SGP Baltík 3. *SGP Systems: Baltík C# 3D vizuální výukové programovací nástroje pro děti, mládež a dospělé* [online]. [cit. 2017-07-07]. Dostupné z: https://www.sgpsys.com/cz/product_B3.asp

[27] Startuje projekt PRIM [online]. [cit. 2018-04-20]. Dostupné z: http://wvc.pf.jcu.cz/ki/?article=/aktuality/startuje-projekt-prim.html

[28] STEPHENSON, Chris a Valerie BARR. Defining Computational Thinking for K-12. CSTA Voice. 2011, vol. 7, no. 2, pp. 3–4.

[29] *STRATEGIE DIGITÁLNÍHO VZDĚLÁVÁNÍ DO ROKU 2020*. In: . Dostupné z: http://www.vzdelavani2020.cz/images_obsah/dokumenty/strategie/digistrategie.pdf

[30] Tematické celky informatických výukových aktivit. [ZPRACOVAL KOLEKTIV ŘEŠITELŮ POD VEDENÍM VLADIMÍRA RAMBOUSKA]. Výzkum informační výchovy na základních školách. Plzeň: Koniáš, 2007, s. 215-238. ISBN 8086948102.

[31] Tvořivost. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-04-20]. Dostupné z: https://cs.wikipedia.org/wiki/Tvořivost

[32] VITOVSKÝ, Antonín. *Moderní slovník softwaru*. První vydání. Praha: AV software, 2006. 588 s. ISBN 80-901428-8-5.

[33] WING, Jeannette M. *Computational Thinking: What and Why?* [online]. 2010 [cit. 2013-10-28]. Dostupné z: https://www.cs.cmu.edu/~CompThink/papers/TheLinkWing.pdf

[34] WIRTH, Niklaus. *Algoritmy a štruktúry údajov*. 2.vyd. Bratislava: Alfa, 1989. ISBN 80-05-00153-3.

[35] ŽÁK, Petr. Kreativita a její rozvoj. Brno: Computer press, 2004. 315 s. ISBN 80-251-0457-5. Kapitola Systémy kreativního řešení problému, s. 124-126

17. Internetové zdroje obrázků

[1] *Obrázek 13 - Ukázka programového prostředí SCRATCH: foto autor*

[2] *Obrázek 14 - Baltík - režim Čaruj scénu* [online]. [cit. 2018-06-25]. Dostupné z: https://upload.wikimedia.org/wikipedia/commons/5/5d/Baltik3-screenshot4.PNG

[3] *Obrázek 15 - Baltík - ovládací panel: Programuj pokročilý* [online]. [cit. 2018-06-25]. Dostupné z: *h*ttp://www.programovani.gyholi.cz/files/baltik/matematicke_operace.png

[4] *Obrázek 16 - KODU Game Lab - detail programovacího prostředí: foto autor*

[5] *Obrázek 17 - CUBETTO- detail programovacích dílků* [online]. [cit. 2018-06-25]. Dostupné z: http://m-o-n-o.com/monolith/wp-content/uploads/2016/03/Cubetto_4.jpg

[6] *Obrázek 18 - Programovací prostředí* [online]. [cit. 2018-06-25]. Dostupné z: https://lego.zcu.cz/web/images/stories/nxt_software/app/printscreen_nxt_programming.png

[7] *Obrázek 19 - SmartHub s napájecí jednotkou* [online]. [cit. 2018-06-25]. Dostupné z: https://le-www-live-s.legocdn.com/images/423923/live/sc/Products/45301/45301_Prod_02/950656655854566e0491a5df982c5db1/05ce4f6d-7040-4aa8-8e5c-a56200c62d71/original/05ce4f6d-7040-4aa8-8e5c-a56200c62d71.jpg?fit=inside|855

[8] *Obrázek 20 - Vývojové prostředí WeDo 2.0: foto autor*

[9] *Obrázek 21 - Detail podvozku ozobota s fotosenzory* [online]. [cit. 2018-06-25]. Dostupné z: http://robodoupe.cz/wp-content/uploads/2029/10/IMGP0127b-400x300.jpg

[10] *Obrázek 22 - Ukázka ozokódu tvořený barevnou sekvencí* [online]. [cit. 2018-06-25]. Dostupné z: https://stevebrophy.files.wordpress.com/2015/08/screen-shot-2015-08-15-at-3-39-29-pm.png?w=768&h=549

[11] *Obrázek 23 - Programové prostředí OzoBlockly k programování robota* [online]. [cit. 2018-06-25]. Dostupné z: https://stevebrophy.files.wordpress.com/2015/08/screen-shot-2015-08-15-at-4-12-00-pm.png

[12] *Obrázek 24 - Detail na ovládací prvky* [online]. [cit. 2018-06-25]. Dostupné z: http://mathfour.com/wp-content/uploads/2014/10/BeeBotCommands.jpg

18. Přílohy

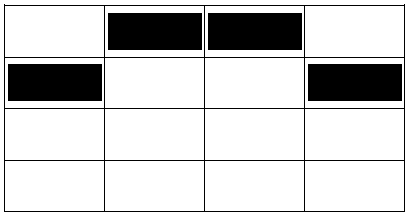
Příloha č. 1a – Sestavy a dohodnuté symboly

Příloha č. 1b – Ukázka možného řešení

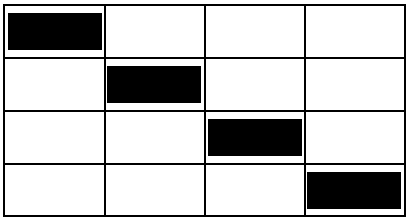
Příloha č. 1c – Prázdné předlohy (hrací kartičky)

Příloha č. 2 – Počáteční a cílová pozice kelímků

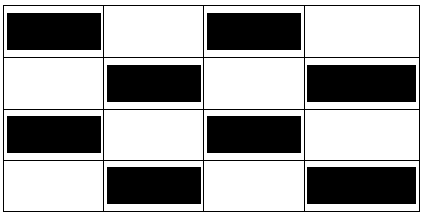
Příloha č. 1a - Sestavy a dohodnuté symboly



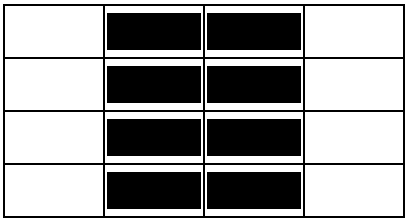
Pozice 1



Pozice 2



Pozice 3



Pozice 4

Dohodnuté symboly:

- ➔ o jedno místo doprava
- ➔ o jedno místo dolu
- ← o jedno místo poleva
- ● vyplnit barvou
- ↑ o jedno místo nahoru

Příloha č. 1b – Ukázka možných řešení

Součástí řešení je také rozšíření o cykly, kde je ovšem nutné předem dohodnout zápis.

Pro tento zápis jsem zvolil dvě varianty, a to variantu pro opakování jednoho příkazu, v tom případě se před příslušný symbol vloží číslovka, značící počet opakovaní a pro skupinu příkazů, kde je tato skupina ohraničena závorkami a opět se před závorku napíše počet opakování.

Pozice 1:

(startovní buňka je vlevo nahoře)

↓ o → ↑ o → o → ↓ o

Jelikož se určitá sekvence zápisu opakuje, lze toho využít pro zavedení opakování (cyklu)

↓ o → ↑ 2(o →) ↓ o

Pozice 2:

(startovní buňka je vlevo nahoře)

o → ↓ o → ↓ o → ↓ o

Jelikož se určitá sekvence zápisu opakuje, lze toho využít pro zavedení opakování (cyklu)

3(o → ↓) o

Pozice 3:

(startovní buňka je vlevo nahoře)

o → → o → ↓ o ← ← o ← ↓ o → → o → ↓ o ← ← o

Jelikož se určitá sekvence zápisu opakuje, lze toho využít pro zavedení opakování (cyklu)

2(o 2 → o → ↓ o 2 ← o ← ↓)

Pozice 4:

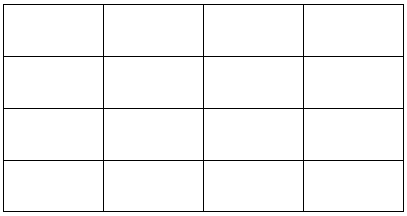
(startovní buňka je vlevo nahoře)

→ → o ↓ o ↓ o ↓ o → o ↑ o ↑ o ↑ o

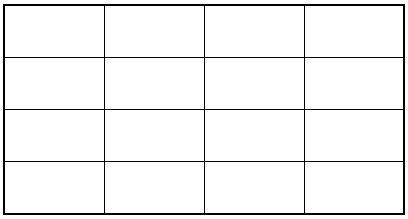
Jelikož se určitá sekvence zápisu opakuje, lze toho využít pro zavedení opakování (cyklu)

2 → 3(o ↓) o → 3(o ↑) o

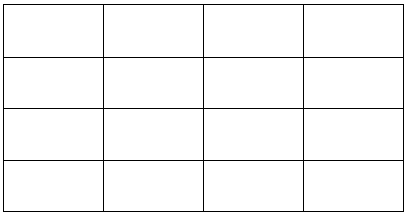
Příloha č. 1c – Prázné předlohy:



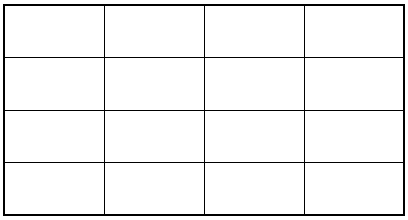
Pozice 1



Pozice 2



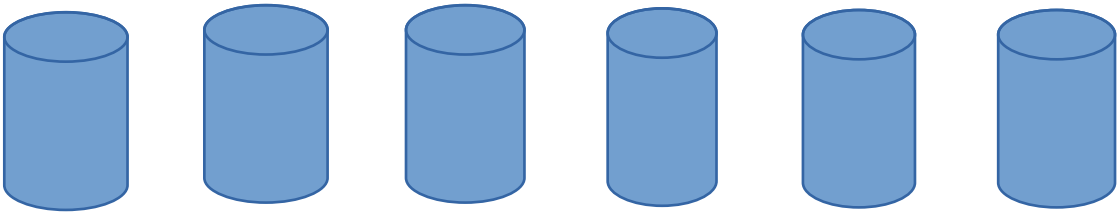
Pozice 3



Pozice 4

Příloha č.2 – Počáteční a cílová pozice kelímků

Počáteční pozice kelímků



Cílová pozice kelímků, kterou mají za úkol žáci dosáhnout.